ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

ANDRÉ DA SILVA ANTONITSCH

TOWARDS A MULTI-LEVEL CROWD SIMULATION MODEL

Porto Alegre

2021

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
GRADUATE PROGRAM IN COMPUTER SCIENCE

TOWARDS A MULTI-LEVEL CROWD SIMULATION MODEL

ANDRÉ ANTONITSCH

Dissertation presented as partial requirement for obtaining the degree of Master in Computer Science at Pontifical Catholic University of Rio Grande do Sul.

Advisor: Prof. Soraia Raupp Musse

Porto Alegre
2021

Ficha Catalográfica

A635t Antonitsch, André da Silva

Towards a Multi-level Crowd Simulation Model / André da Silva Antonitsch. – 2021.
93 p.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientadora: Profa. Dra. Soraia Raupp Musse.

1. Simulação de Multidões. 2. Mobilidade Populacional. I. Musse, Soraia Raupp. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecária responsável: Clarissa Jesinska Selbach CRB-10/2051

André da Silva Antonitsch

Towards a Multi-Level Crowd Simulation Model

This Master Thesis/Doctoral Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor/Master of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 30, 2021.

COMMITTEE MEMBERS:

Prof. Dr. Nuria Pelechano (CGVR/UPC)

Prof. Dr. Rafael Heitor Bordini (PPGCC/PUCRS)

Prof. Dr. Soraia Raupp Musse (PPGCC/PUCRS - Advisor)

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

TOWARDS A MULTI-LEVEL CROWD SIMULATION MODEL

RESUMO

Em geral, modelos de simulação de multidões tem níveis-de-detalhe predefinidos, sejam microscópicos, macroscópicos ou uma combinação de dois modelos, como modelos híbridos. Modelos macroscópicos oferecem um *trade-off* entre acurácia de simulação (ou individualidade dos agentes) e complexidade da simulação, quando comparados com modelos microscópicos. Modelos híbridos equilibram esse *trade-off* ao definir regiões no ambiente a serem simuladas com cada estratégia, definindo regiões de mais ou menos importância. Este trabalho foca em modelos capazes de adaptar o nível-de-detalhe dinamicamente. Propomos novos modelos, BioClouds e Legion, que introduzem os conceitos de discretização e competição de espaço em modelos de macroscopios de simulação de multidões. O *Level-Of-Detail Urban Simulation framework* é um *framework* feito para descrição de cidades com nível-de-detalhe dinâmico que contém dois modulos, uma ferramenta de geração de cidades e um modulo de simulação de mobilidade populacional. O presente trabalho apresenta a simulação de mobilidade populacional do LODUS, que utiliza de estratégias de simulação de multidões para solucionar tarefas de mobilidade populacional. Nós propomos três casos de uso para o LODUS: mobiilidade urbana, transporte público e contágio e propagação de doenças. Nós realizamos experimentos visando analisar qualitativamente comportamentos emergentes para o cenário de uma simulação de mobilidade urbana.

Palavras Chave: Simulação de Multidões, Mobilidade Populacional.

TOWARDS A MULTI-LEVEL CROWD SIMULATION MODEL

ABSTRACT

In general, crowd simulation models have pre-defined level-of-detail, be it microscopic, microscopic or a combination of two models, like hybrid models. Macroscopic models offer a trade-off between simulation accuracy (or individuality of agents) and simulation complexity, when compared with microscopic models. Hybrid models try to offset this trade-off by defining regions in the environment to be simulated with each strategy, defining regions of more or less importance. The present work focuses on models capable of shifting level-of-detail dynamically. We propose BioClouds and Legion, which are novel macroscopic crowd simulation models, introducing the concept of space discretization and competition to macroscopic crowd simulation. The Level-Of-Detail Urban Simulation framework is a framework to describe cities with a dynamic level-of-detail which contains two modules, a city generation tool and a population mobility simulation module. The present work presents the LODUS population mobility simulation, which uses a crowd simulation approach to solve the task of simulating population mobility. We propose three use cases for the population mobility simulation: a urban mobility scenario, public transportation and disease spreading. We performed experiments to qualitatively assess emergent behavior for the urban mobility scenario.

Keywords: Crowd Simulation, Population Mobility.

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

Crowd simulation solves tasks where experimenting or gathering with a large number of real people is unfeasible, for example, finding out the best evacuation plan for a building becomes unreasonable if you have to physically move people in and out of buildings, or animating a huge battle scene for a movie because it is prohibitively expensive. This applies to both very small, refined crowds and very large crowds of people where individual features are indistinguishable. Crowd simulation has been studied and investigated in last 30 years [22] and still some challenges exist. One of these challenges is related to the generalization of crowds at different levels of control, individual, group and large crowds.

Microscopic and macroscopic simulations differ in how they model entities which represent crowds. Microscopic simulation models are good fits to simulate small crowds with individual behavior. There exist several models which fit this category, such as, BioCrowds [5], ORCA [24] and the social forces model [11]. BioCrowds [5] is a space discretization simulation model which guarantees collision avoidance between its agents by enforcing a convex partitioning of space by the participating agents. ORCA [24] simulates collision avoiding crowds by computing the mutual interaction of velocity obstacles, that is, the future positions each agent will occupy. Social forces [11] conceptualizes agents as being affected by internal and external factors, leading to emergent behavior.

Macroscopic simulation models are good fits to simulate large abstract crowds where behavior is homogeneous, i.e., not individualized. Some models which fit this category are BioClouds [2], a space discretization model, fluid dynamics models [10] and continuum crowds [23]. Hybrid simulation models were created to balance the computational efficiency of macroscopic models and the individual behavior and interactions of microscopic models. Xiong et al. [25] propose combining a microscopic and a microscopic models to simulate different regions of an environment with different levels-of-detail.

In this work, we present two models to further explore the idea of levels-of-detail in crowd simulation. Firstly, Legion, which is a multi-level behavioral model based on space discretization approach; and secondly, a population mobility framework to generate large scale simulations. Legion [2] is a crowd abstraction based on BioClouds [1], capable of dynamically adapting to an environment of different complexities. Legions, differently than clouds, from BioClouds, can split and merge into groups of different sizes, when faced with obstacles and open areas. Secondly, we present the *Level-of-Detail Urban Simulator* (LODUS) a framework composed of a procedural city-generation tool and a variable-scale population mobility simulation. The present work proposes the population mobility simulation model to be used in LODUS. The original LODUS work is presented by Silva et al. [21].

The LODUS population mobility simulation framework is a framework design to allow simulation designers to generate simulations involving movement of large scale populations of any desired scale, i.e. cities, states, countries and so on. The framework is data-driven, adapting its representation of a population to the available data. With an additional set of operations we seek to model the urban mobility of a city, such as Porto Alegre. We base the framework on the idea proposed by Gonzalez et al. [8], that human mobility patterns are repetitive over time. In this way, we model an environment as a set of points-of-interest, and the population as resources to be traded regularly by those locations. Our population representation expands on the idea of having a cloud of agents proposed in BioClouds [2], where clouds represent a region occupied by similar agents. The population model for the population mobility framework (LODUS) uses the terminology *blob* of agents, i.e., a group of agents with a certain population profile, which can be separated and merged back together.

Using the LODUS population mobility framework, we designed a few experimental scenarios based on the COVID-19 pandemic during the year of 2020, using real-word mobility data for the city of Porto Alegre. We also demonstrate possible models to simulate different situations, public transportation with multiple transport modalities (bus, individual transport) and a simplified possible model to simulate disease spreading using population mobility as a part of the simulation.

By combining models to increase and decrease the level-of-detail of a macroscopic simulation we can build a new approach which can simulate from the singular agent up to the largest population. We build upon that idea by presenting the Legion model, which individualizes clouds up to the individual level, and a blob based population mobility framework, which can be translated into Legion simulations for a given point-of-interest. As far as we know this is the first model to connect population mobility in a macroscopic model with a microscopic model, allowing to simulate individuals.

This work is organized as follows: Section 2 goes into details of related work in the fields of crowd simulation and population mobility. Sections 3, 4 and 5 present the three models proposed during this master's degree, BioClouds, Legion and the population mobility module of LODUS, respectively. Section 6 presents experimental results for BioClouds, Legion and LODUS. Finally, Section 7 presents our final remarks and debates future work for this research.

2. RELATED WORK

This section discusses studies from the following areas of research: crowd simulation and population mobility. Regarding crowd simulation, this work describes existing approaches for microscopic and macroscopic simulation and their relevance to the individualization of individuals in a crowd. The methods presented here are viewed under the optics of how to simulate crowds with the goal of achieving a multi-level simulation. In the topic of population mobility this work presents two statistical models for the description of population mobility and temporal behavior. We also present a crowd simulation model to analyze the impact of environment and mobility on the spread of COVID-19.

The field of crowd simulation is vast and its uses can range from fire drill planning, crime prevention and pedestrian flow analysis [22]. The analysis of how a crowd behaves in an environment as a whole is largely the main concern of the field. This section presents microscopic models of crowd simulation, where each agent is modeled as a single entity of the simulation, macroscopic models, where each simulation entity represents several agents, or agents are represented as a field distribution, density cloud or similar, hybrid crowd simulation models, where agents and crowds are presented as a mix of microscopic and macroscopic approaches and, population mobility analysis and simulation, where individuals are completely abstracted and the simulation is more interested in larger scale temporal and spatial characteristics of a population.

2.1 Microscopic Crowd Simulation

Microscopic crowd simulation models each agent in a crowd as a single entity of simulation. These models can vary on the degree each individual simulated agent presents particular behavior in relation to other agents, where homogeneous models provide agents with little or no individuality and heterogeneous models simulate agents with individual behavior.

2.1.1 Homogeneous Microscopic Crowd Simulation

Models such as, the social forces model [11], ORCA [24], and BioCrowds [5] are examples of crowd simulation where agents and homogeneous. Agents will behave similarly and showing little individuality.

Helbing et al. [11] introduced the concept of social forces. The social forces model conceptualizes agent movement as if influenced by internal and environmental characteristics, such as desired velocity, distance from other pedestrians and a term modeling attractive effects of simulation. These simulations show to some self organizing behavior observed in real crowds.

Van den Berg et al. [24] propose the ORCA model, which simulates a crowd based on reciprocal velocity obstacles (RVO). The simulator is solved with linear programming, which was shown to solve a step in O(n) time. The model was originally created for robotics controllers and tends to give little personal space for each agent. The model has extensions that cause it to present more natural crowd forming behaviour, e.g., a simulation which respects the fundamental diagram [17], i.e. a diagram which describe how fast each individual is willing to move in a certain density, used for modeling culturality, personality and etc.

Bicho et al. [5] introduced the space discretization and competition model for crowd simulation. BioCrowds supposes agents are competing to maintain their personal space and to do so will take possession of space around them. To model the environment, the authors propose using space discretization markers. Agents take possession of markers closest to them, forming an emergent Voronoi partitioning of space. The convex Voronoi cells each agent now occupies guarantee collision avoidance, as long as no agent attempts to move outside its cell.

BioCrowds also features self organizing emergent crowd behavior, most notably the emergence of lanes, vortices, bottle neck and arc formation.

2.1.2 Heterogeneous Individual Behaviors

The individual behavior of an agent in the crowd can be modeled beyond the scope of simple collision avoidance. There has been research on how to further customize, or adapt, a crowd behavior to a different set of conditions. Among those, cultural crowd behaviors, thermal comfort and emotion contagion.

Cultural Aspects

Favaretto et al. [7] proposed a model capable of capturing cultural differences of crowds based on video and simulation analysis. The GeoMind model is a geometric analysis of how a group of people, simulated or filmed, occupy a given space. The space occupation of a crowd is associated with the OCEAN personality model. OCEAN parameters measure a person's *Openness*, *Conscientiousness*, *Extroversion*, *Agreeableness* and *Neuroticism*. These values are tracked in videos displaying real life crowds of different cultural backgrounds and spaces, extracting measurable differences between people from different cultures. The goal of GeoMind is to extract geometrical information of a group, that process is divided into two steps: individual characteristics extraction and a mapping of crowd features into *cultural dimensions*. GeoMind analyzes crowd videos or simulations and first extracts individualized information for each in the crowd. The model extracts position, speed, angular speed in relation to a reference vector. Based on those features, the model computes three other measurements: collectivity, socialization and isolation levels for each agent. Based on extracted features, the model computes OCEAN values for each individual. The authors presented data which indicate the computed OCEAN values are dissimilar between crowds of different cultures.

Narang et al. [17] propose an extension to the ORCA [24] model to generate trajectories which display the density to speed relationship of the Fundamental Diagrams of a given crowd. The DORCA model proposes the use of density filters to map the densities and occupations of an environment. The density filters acts as an interface between a global planner and agent local path planner. The agent will seek to optimize its preferred speed to conform with the information provided by the density filter. The authors present results which show DORCA is conforms more, in four different environments, with a real life crowd's Fundamental Diagram than the original ORCA.

Psychological Aspects

Pelechano et al. [20] propose a model to incorporate psychological aspects and communication into crowd simulations. The PMFserv system, integrated with the crowd simulation system MACES, models human behaviour like stress, emotion, motivation, decision process and ability. The PMFserv uses performance moderator functions, grounded on real-life data, and human behavior representations to model decision-making under varied agent emotional state and subjective utilities. The decision making can be further constrained by stress levels and physiological state. The work proposes modeling micro-behaviors for agents aiming to generate emergent cultural behavior.

Durupinar et al. [6] propose a model to differentiate, in a simulation, two types of crowds: audiences and mobs. The authors define audiences as passive crowds and mobs as active crowds, influenced by emotional and homogeneous behavior. The work has the following main contributions: introduces an emotion contagion model based on OCEAN personality traits; employs the OOC emotion model in multi-agent simulations; and utilizes a pleasure-arousal-dominance model do influence agent decision making based on emotion expression and behavior selection. The authors present scenarios simulating situations of protesters and police interactions and a sale event with agents rushing towards sales.

Neto et al. [19] propose a model to simulated the spread of psychological phenomena in a crowd of virtual agents. The model incorporates psychological characteristics of humans into agents to simulate a model of emotional contagion in a crowd scenario. The model is further associates OCEAN personality traits with agents in a crowd simulation. The model adds an emotion information to agents in a BioCrowds simulation and provides the agents with the ability to spread that information with other agents. At a given simulation step t, the emotion contagion is affected by a number of factors, e.g., each agents position, expressiveness, susceptibility to contagion, current goal and to which group each agent participating in contagion belongs to.

2.2 Macroscopic Crowd Simulation

Macroscopic simulation of crowds represents each agent as part of a collective, where the control is on the macro-structure level [22]. These collectives of agents can be represented as density fields or an aggregation of agents. Some examples of these models are the Continuum crowds model and the BioClouds model.

Treuille et al. [23] proposed Continuum Crowds, a crowd simulation model based on continuum field dynamics. The model simulates a crowd as a dynamic potential field to perform agent navigation for agents and moving obstacles. Continuum crowds models crowds without agent-based dynamics.

The continuum crowds is modeled according to four hypothesis:

- Each person is trying to reach a geographic goal;
- People move at the maximum speed possible;
- There exists a discomfort field which determines location preferences;
- Agents attempt to minimize path length, discomfort and time.

The system takes a dynamic potential approach to generate velocity fields. Entities perform path planning to avoid collisions.

The BioClouds model will be described in this work, presented in depth in Section 3.

2.3 Hybrid Crowd Simulation

Xiong et al. [25] propose the idea of a hybrid approach for crowd simulation, separating the simulation environment into a microscopic and a macroscopic simulated regions. The authors propose leveraging the accuracy and computational efficiency of both models. The work uses the ORCA model as a microscopic simulation model, restricted to a region of interest, and the remaining space is simulated using the continuum crowds model. The regions are interfaced with a transition region. When an agent crosses the boundary, it is removed from one simulation and added to the other. The regions in the work are defined during the environment description, and are static during simulation.

2.4 Population Mobility

Population mobility abstracts key aspects of a population from a set of parameters to be modeled, where the notion of the individual is, in general, non-existent. This is usually a statistical approach to large scale temporal human mobility and migration patterns. In recent times the COVID-19 epidemic sparked renewed interest in the study of urban mobility and its effects on disease spreading. Proposed methods such as Jamshidi et al. [14] and Hernández-Orallo et al. [12] present simulation studies on the impacts of mobility, mask usage and mobility models in the study of COVID-19, while Brockmann et al. [4] and Gonzalez et al. [8] present analytical studies on the characteristics of human mobility in urban environments

Brockmann et al. [4] and Gonzalez et al. [8] propose that human mobility patterns follows a power law. Brockmann reached that conclusion by tracking bank notes serial numbers. Gonzalez et al. used temporal data of cellphone location to derive a pattern of population mobility in a city. Their models found temporal similarity in all patterns of human mobility. The authors argue this temporal similarity impact human mobility among other aspects. One of the key findings of these studies is the finding that humans mostly visit a small set of a few favorite locations repeatedly as days passes with a random chance to visit novel locations each day. The authors also concluded that humans in a city tend to travel the most to places close to where they currently are, with fewer longer jumps. This behavior was modeled as a Lévy Flight [15], a random walk with distances sampling a Lévy distribution.

Jamshidi et al. [14] present an evaluation of the effect of the weather on human mobility, urban density and social isolation with and the subsequent effect on the spread of COVID-19. The work highlights the effects of spatial and temporal scale on the infection models for COVID-19. Hernández-Orallo et al. [12] explored the use of urban simulation to evaluate the temporal and spatial risks of infection in a urban scenario. The explored situations were as such as train stations and public squares, and the authors generated heatmaps to locate areas of increased risk. The goal is to possibly help health agencies to mitigate risks in public spaces.

2.5 The Present Work in the State-of-Art

Regarding the presented context of the state-of-art, this work aims to expand on the idea of multi-level models. That is, models which can dynamically shift between different degrees of of microscopic and macroscopic simulation. This behavior is similar to hybrid crowd simulations, where a simulation is pre-defined to use two simulation models on static regions and conditions, while on multi-level simulations those conditions are dynamically defined by the user and simulation current conditions. A microscopic model for regions of particular interest, and a macroscopic model for the rest of the simulation, where simulation accuracy is not the main concern of the simulation.

In this way, BioClouds, Legion and LODUS present steps to create a crowd simulation framework which can perform multi-level crowd simulation. BioClouds presents this idea with a multi-simulation approach, using a BioCrowds simulation to provide individualized agent visualization. Legion provides a dynamic hybrid crowd simulation, with simulation entities, legions, which can split and merge depending on environment complexity, opposed to work like Xiong et al. [25] which present static regions to define simulation granularity. LODUS expands on the ideas of using crowd simulation strategies and groups of individuals of varying sizes to simulate a population mobility scenario, with regions of arbitrary sizes, like cities, states or countries.

The three methods proposed in this dissertation are detailed in next chapters.

3. BIOCLOUDS

BioClouds model [2] is a generalization of BioCrowds [5], and as such, is a space discretization and competition model for collision avoidance of crowds. BioClouds models entities as a group of agents, called a *cloud* and space markers as a space partitioning of space, that is, cells containing a certain area of space.

The BioCrowds model is a space discretization crowd simulation algorithm in which agents compete for discretized space. This space competition is performed over a discretization of the simulation environment, allowing agents to interact with free available space and dispute personal space. Next section briefly details BioCrowds in order to provide a better understanding of BioClouds and Legions models.

3.1 The BioCrowds Model

BioCrowds method [5] proposes the environment discretization with uniformly distributed markers. Agents in the environment compete for those markers, based on proximity criteria, and use them to determine their movement vectors. Indeed, each agent i accesses the markers inside its personal space R_i to search for markers that are closest to i than any other agent j. So, a marker is only available to the closest agent. For a given agent i, with a set of N available markers denoted by $S = \{\mathbf{a_1}, \mathbf{a_2}, \cdots, \mathbf{a_N}\}$, we calculate its movement vector $\mathbf{m}$ using Equation 3.1:

$$\mathbf{m} = \sum_{k=1}^{N} w_k(\mathbf{a_k} - \mathbf{X}), \tag{3.1}$$

where $\mathbf{a_k}$ is the marker's position and $\mathbf{X}$ is the agent's position. The marker's weight w_k is calculated from Equation 3.2:

$$w_k = \frac{f(\mathbf{g} - \mathbf{X}, \mathbf{a_k} - \mathbf{X})}{\sum_{l=1}^{N} f(\mathbf{g} - \mathbf{X}, \mathbf{a_l} - \mathbf{X})}, \tag{3.2}$$

where $\mathbf{g}$ is the goal position of agent i. Function f should prioritize markers that lead the agent directly to its goal. A possible choice is defined in Equation 3.3:

$$f(x, y) = \frac{1 + cos\theta}{1 + ||y||}, \tag{3.3}$$

where θ is the angle between x and y. The model should allow the agent to move with a maximum desired speed S_{max}. However, in dense crowds, the space available for each agent is smaller, resulting in a speed reduction. Therefore, in the proposed model, the instantaneous motion vector can be defined by:

$$v = s_{min}\frac{\mathbf{m}}{||\mathbf{m}||}, \tag{3.4}$$

where $s_{min} = min(||\mathbf{m}||, S_{max})$ which implies that if $||\mathbf{m}|| > S_{max}$, the maximum displacement is limited by S_{max}. Otherwise, it is given by $||\mathbf{m}||$. Please refer to BioCrowds original paper [5] for further details about the method.

3.2 From BioCrowds to BioClouds Model

The cloud's behaviors differ from a BioCrowds agent in the following aspects: the space capturing radius and the distance function used for space competition. Space competition radius is determined by a desired density, a quantity of agents and a radius factor. This radius factor increases the desired radius as the cloud increases captured space density. Clouds use the power-of-a-point distance function [3] , which generates power diagrams, a generalization of Voronoi diagrams. The power diagrams display the mathematical properties necessary by the BioCrowds collision avoidance guarantees. In order to model a crowd's desire to maintain a certain density, *clouds* possess a perception radius balancing behaviour: a *cloud* which is under its preferred density will shrink its radius to better accommodate its desires. Similarly, a *cloud* which is over its desired density will grow its radius in an attempt to capture more space.

The movement of a cloud, owner a set of captured space markers, is given by the same equations as the BioCrowds model. The model differences offer an exchange between accuracy and simulation complexity, with a simulation on the same granularity of BioCrowds being also possible.

Space markers are used to discretize the space and are defined as an uniform grid of square cells. In this work, it is used cells of area= $4sqm$ for cells of side= $2m$. The cells can be owned by, at most, a single cloud of agents at a time, in order to provide collision avoidance. Obstacles can be simulated as an absence of cells in a given region of space, so *clouds* will not be able to mark such regions as owned by them and, therefore, are not going to move into that region. The area that cells represent is a parameter of the model and can be adjusted as intended.

Clouds, in its turn, are aggregations of agents which have common characteristics and goals. The number of agents a certain cloud C_i contains is one of the could parameters. Indeed, the cloud is a structure that contains a set of parameters, defined as follows for cloud i: $C_i = \{A_i, dD_i,$ $dS_i, dA_i, R_{i,t}, sR_i, \vec{g_i}, jD_{i,t}, jA_{i,t}\}$, where A_i states for the number of agents inside cloud i, dD_i is the desired density the cloud should achieve, dS_i is the desired speed and dA_i is the desired area such cloud wants to occupy in order to keep the desired density. In addition, $R_{i,t}$ is the cloud radius at instant t, estimating that its shape is a circle[1]. This parameter states for the radius of the perception area where cloud i is going to search for space markers (i.e. cells in BioClouds) in the environment. Clouds can update their $R_{i,t}$ radius parameter if instantaneous density is different from desired one. Such update is defined by a radius change speed sR_i parameter, which determines how much the perception radius can change in a single simulation step, the maximum change is defined by dS_i,

[1]In order to provide a collision free algorithm, we consider that our clouds should be represented by convex polygons, so in order to avoid extra computation, it is considered as a circle only local density purposes.

as later described in Equation 3.6. In addition, $\vec{g_i}$ states for cloud i goal. Finally, the instantaneous values $jD_{i,t}$ and $jA_{i,t}$ are the instantaneous density and area, respectively, at instant t.

The main behavior of the cloud is to move towards its goal while trying to keep its desired structure (density and velocity) and avoid collision with obstacles and other clouds. It implies that the cloud i tries to keep its preferred density parameter dD_i which, in its turn, is related to the cloud area dA_i. Once the number of agents A_i and the desired density dD_i are defined for a cloud C_i, the desired cloud area is computed such as: $dA_i = \frac{A_i}{dD_i}$, as well the initial radius of the circular shape to contain the cloud: $R_{i,0} = \sqrt{\frac{A_i}{dD_i \Pi}}$. Therefore, C_i will try to keep its desired density during the motion to achieve the goal $\vec{g_i}$. However, due to space constraints (for example, crowded situations), clouds can lose space (i.e. cells) to other clouds. Such behavior diminishes the instantaneous area ($jA_{i,t}$) occupied by C_i at instant t, which can increase the instantaneous density ($jD_{i,t} = \frac{A_i}{jA_{i,t}}$). In fact, $jA_{i,t}$ and $jD_{i,t}$ are computed at each frame and depends on the area (in cells) that the cloud is occupying at the moment.

Once the clouds in BioClouds can have different amount of agents and different densities, their radii can be different, so the Euclidean distance function used to attribute markers to clouds is not appropriate anymore as it is in BioCrowds for agents and markers. Indeed, in BioCrowds, the Euclidean distance is used to decide if a certain marker should be attributed to a determined agent, since markers are attributed to the closest agent. However in BioClouds, clouds can have different radii so we propose to use another distance function in order to provide a correct way to attribute the markers to the clouds, allowing free-of-collision clouds movement. To do so, the Power of a Point distance [3] method was chosen, because it partitions the space into an emergent Voronoi-like diagram, so it keeps the method collision-free. To clarify, the power of a point distance function is defined in Equation 3.5:

$$Pd_{i,k,t} = d(\vec{X}_{i,t}, \vec{a}_k)^2 - R_{i,t}^2, \tag{3.5}$$

where $\vec{X}_{i,t}$ is the center position of cloud C_i at frame t, $\vec{a}_k$ states for the position of a given marker k that the cloud C_i is computing the distance, function $d(,)$ denotes the inner product and $R_{i,t}$ is the estimated C_i radius at frame t. Considering the relation between the power of a point distance weight and the radius of a circle, the area a cloud occupies is approximated to that of a circle. Considering that clouds can lose space, increasing instantaneous density, when competing for space, R_i must be updated in a next frame to reflect how the cloud shape can change to accommodate agents, and at the same time limit how compressed a cloud can possibly be. $R_{i,t+1}$ is then computed according to the ratio between its instantaneous density $jD_{i,t}$ and the preferred density dD_i, as seen in Equations 3.6 and 3.7:

$$R_{i,t+1} = R_{i,t} + min(dS_i, max(-dS_i, sR_i\ \beta_t\ R_{i,t})), \tag{3.6}$$

where β_t can be defined as shown in Equation 3.7:

$$\beta_t = min\left(\frac{jD_{i,t}^2}{dD_i^2}, 2\right) - 1. \tag{3.7}$$

The *clouds* movement vector equation is the same as the one defined for BioCrowds (see Equation 3.1), and, as such, possess the same properties of only moving a cloud inside its own emergent Voronoi polygon. The emergent Voronoi diagram of space associated with the movement equation ensures that the *clouds* never collide between themselves. Unlike BioCrowds, the cells in BioClouds have an area value, so a *cloud*'s area can be measured by the total sum of owned cells areas.

BioCloud model was implemented using the *Unity3D* framework[2]. It was built to allow for the parallel computing of each step. The simulation performs the following steps:

1. Cells choose the closest cloud in the same way agents do with markers in BioCrowds;
2. Each cloud uses its owned cells to compute a resulting movement vector (Equation 3.1);
3. Each cloud updates its current radius (Equation 3.6); and finally
4. Each cloud moves according to its resulting movement vector.

[2]https://unity3d.com/

4. LEGION

Legion is a multi-level crowd simulation model [1], i.e., the simulation dynamically changes the level-of-detail of the modeled agents. Legion also introduced a blobby model visualization approach to render dynamically defined blobs of agents, i.e., a visualization model capable of representing crowds no matter the current level-of-detail of the simulation. Legions are similar to BioClouds clouds, in which each legion models a group of individuals with similar goals, behaviors and characteristics. The main difference between both models is the ability legions have to split and merge when navigating an environment with varying degrees of complexity (obstacle quantities, path shapes and so on). This leads to a simulation which is dynamically multi-level, offsetting computational complexity and simulation accuracy (when compared to simulating agents individually) as required by the user.

The blobby visualization presents a more organic look than the one presented in BioClouds, where clouds are rigidly shaped. The blobby model represents legions and the environment as distance functions. As such, blobby visualized legions can change shape according to their sizes, proximity with other legions and proximity with obstacles.

Indeed, *Legion* is a new macroscopic crowd abstraction which purpose is to address the multi-level behavior, allowing the simulation of groups of varied sizes that react to the complexity of the environment. Legions represent crowds of people without simulating individuals, as proposed in BioClouds model [2]. We introduce the possibility of having different levels between the higher level (macroscopic) and the individual agent (microscopic) in an emergent way. The Legion can be subdivided into crowds, smaller crowds, large groups, smaller groups, and successively down to the level of individuals. The subdivision of vast legions into smaller groups of various sizes occurs depending on the environment.

The main contribution of our model is that it can deal with those varied sized groups as a single entity, keeping the group properties as goals, speed, and density as if it were individual behaviors. Moreover, the subdivision can be followed by group merging, and it happens only depending on the space restrictions. Unlike the approach proposed by Xiong et al. [25] of defining specific regions where the multi-level behavior can be achieved, in our model, the split and merge behaviors are dynamically adapted to the complexity of the environment. We also seek to represent our legions organically, compared to the usual methods to visualize macroscopic crowds [2], [13], as discussed in Section 3.

4.1 Multi-level control

In real life, large crowds can be divided into smaller groups to get around obstacles. Indeed, smaller groups accommodate better in free space when competing for regions with other groups.

After a crowd overcomes an obstacle, groups can merge back into a single large crowd, if there is free space.

Our legion model uses BioCrowds [5], as BioClouds [2], which means that we also use space markers to provide motion to our legions (as explained in Section 3). It is essential to note that, The original BioClouds model proposed increasing the cloud's radius when free space decreases so that the cloud can use more markers on the ground. Here, we intend to model these situations in real life, allowing the best efficiency of the macroscopic simulation, while preserving the more precise behaviors of groups with fewer individuals. Next sections present split and merge behaviors.

4.1.1 Splitting

Legion splitting occurs whenever a legion is in a situation (time t) where, in a future moment (time $t+1$), it might have less space to accommodate the Legion than currently desired. For example, when a group moves towards an obstacle, that obstacle takes up space, the group cannot occupy, thereby reducing its future space availability.

Each legion i has following attributes: the number of agents it represents (A_i); the desired density (D_i in people/m^2); speed (S_i in m/s); legion goal $\vec{G_i}$; and a perception radius (R_i) equal to the corresponding radius to guarantee the desired density D_i. In addition, as in BioClouds [2], the entities evolve in a discretized space represented as a regular grid, which cells, $M_{i,t}$, are possessed by the legion i at each frame t. As long as the group has the same amount of agents, the same density and space desired to accommodate the structure, R_i, will not be changed. In a situation of division, that is, the space needed in the $t+1$ is less than the desired for the crowd to move. This structure is subdivided into 7 new structures in the location of its current area, 6 radial legions and a central one, in a hexagonal pattern. Six from the new 7 legions are located at the edge of the original and are more likely to capture space than the single cloud. The radius of the newly created clouds is one-third of the original radius of the clouds. Figure 4.1 shows a diagram of the cloud's split pattern. The radial arrangement was chosen to facilitate the creation of smaller clouds in a pattern that best mimics the original density and space occupation. Although we chose a seven-way split for this work, the number of clouds created during a split event can be adapted to better simulate each possible experiment.

The decision to split a legion i at time $t+1$, with A_i agents, position $\vec{X}_{i,t}$, set of captured cells $M_{i,t}$, movement vector $\vec{V}_{i,t}$ and perception radius R_i is based on the available space for the legion i at frame $t+1$. We check for future available space by projecting the future position of the legion $\vec{X}_{i,t+1}$:

$$\vec{X}_{i,t+1} = \vec{X}_{i,t} + \vec{V}_{i,t}. \tag{4.1}$$

Then we check if the markers[1] within radius R_i around $\vec{X}_{i,t+1}$ are in the set $M_{i,t}$ or free (i.e., do not attributed to any other legion or obstacle). If this condition is true, no split is necessary. To avoid computing the split decision, at each frame, we do it at every f frames. We have found $f = 15$ produces good visual results.

If a split happens, each one of the newly created legions (in present case 7 legions) receives a new ID and inherits the same velocity, goal, and desired density as the parent legion. Moreover, R is calculated based on the number of agents assigned to each Legion ($1/7$ of parent legion) and the desired density. The set of captured cells M_t and the motion vector $\vec{V}_t$ are calculated to provide the movement of that Legion at frame $t+1$ guiding to goal $\vec{G}$. Since the cloud division can occur recursively, each structure can be divided again until the resulting structure represents only one individual.

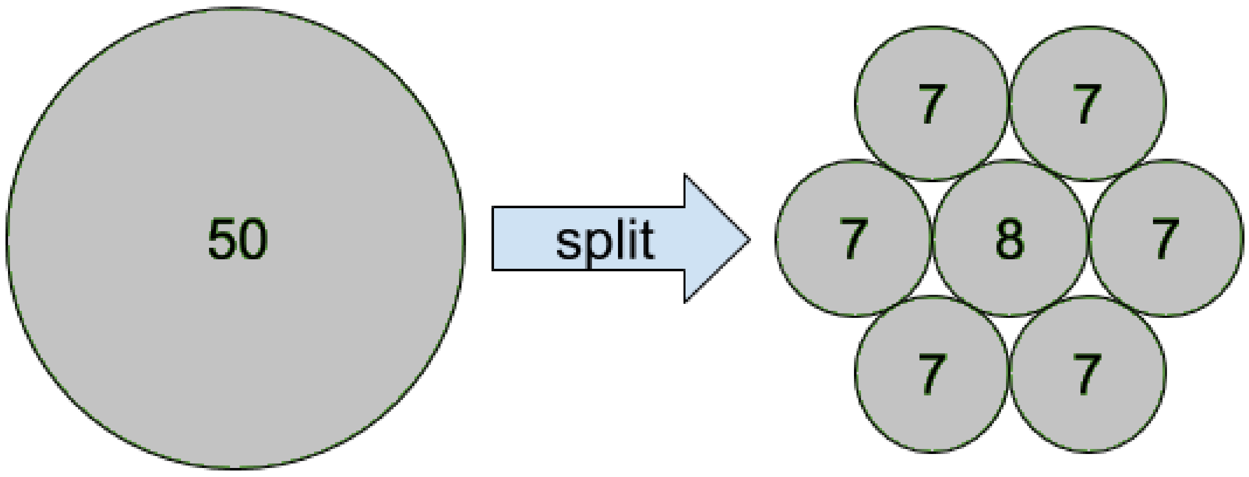


Figure 4.1: A split event: a 50-agent legion splits into 7 smaller legions. The new central legion receives the remaining agents in case of fractional divisions. This pattern arbitrarily chosen, due to its symmetry.

4.1.2 Merging

Cloud merging behavior aims to regroup pair of legions (i and j) when they have the same goals ($\vec{G}_i = \vec{G}_j$) and are close enough to be reunited. We determine the second rule through a simple heuristic: we consider the markers that are close to a straight line from the two clouds centers ($\vec{X}_{i,t}$ and $\vec{X}_{j,t}$); if all such markers are part of subsets M_i or M_j then two clouds are adjacent and can be reunited. If this condition is not true, there is space between the two clouds, not indicating that they are adjacent, so merge behavior is not applied. Legion merging conditions are tested at each simulation step.

[1]Markers are dots on the floor in a simulated environment. They are used in BioCrowds [5], BioClouds [2] and in the current method to discretize the space, but continuous simulation are also possible

A legion that is in a situation that leads to splitting and also merging would cause a not desirable repetition of splitting and merging behaviors. To avoid such sequential operations, we add a settling time St_i for each newly created legion, either by splitting or merging operations. This structure is defined as: $St_i = \{st_i, mt_i\}$, where st_i and mt_i state for last split and last merged frame, respectively. In addition, we define two other thresholds: *shift*$_{split}$ and *shift*$_{merge}$, which state for a number of frames where a new split/merge cannot happen. Such parameters work in a way that only in simulation frame $f = st_i +$ *shift*$_{merge}$, a new merge can occur for legion i. Similar process happens to activate again the split operation, i.e., when simulation $f = mt_i +$ *shift*$_{split}$. In this work we empirically chosen *shift*$_{merge}$ $=$ *shift*$_{split}$ $= 30$ simulation frames because it results in visually coherent simulations. This value is equivalent to 3 seconds of simulation time $(10f/s)$, and can be changed to better fit different environments. When two legions merge, their data is combined to define a new legion with a new ID.

4.2 The control algorithm

At the beginning of the simulation, legions are generated based on user input, who also defines the environment and the obstacles. For each legion i at $t = 1$, the user defines:

1. number of agents A_i,
2. initial position $\vec{X}_{i,t}$,
3. desired density D_i (people/m^2),
4. desired speed S_i (m/s), and
5. goal $\vec{G}_i$ (X,Y,Z) location.

Based on these data, our method computes the perception radius R_i that the legion must have to achieve density D_i, selects the set $M_{i,t}$ with the markers inside R_i that could be attributed to legion i, and finally computes the movement vector $\vec{V}_{i,t}$ based also on speed S_i and goal $\vec{G}_i$.

Each legion is subdivided in 7 other legions when the split behavior occurs. Let us consider legion k as one of the 7 new legions generated from i. The data used to instantiate k is:

1. number of agents (int) $A_k = A_i/7$,
2. initial position
   $\vec{X}_{k,t} = \vec{X}_{i,t} + [(\sin(2\pi k/6), \cos(2\pi k/6)) \times R_i/2]$,
3. desired density $D_K = D_i$,
4. desired velocity $S_k = S_i$, and
5. goal $\vec{G}_k = \vec{G}_i$.

As for legion i, the method computes the perception radius R_k, $M_{k,t}$, and finally $\vec{V}_{k,t}$. The new structures cannot merge during 30 frames, i.e., 3 seconds of simulation time, as defined in Section 4.1.2.

During the simulation, legions evolve in the environment, avoiding collision with other legions and obstacles (when space to move is not free). Therefore, they can split again and successively, until the Legion has only one individual.

For merge situations, as described before, two legions i and j can merge if they have same goals and be adjacent. In this case, their data is gathered to compose a new legion (e.g., m), where data is instantiated as follows:

1. number of agents $A_m = A_i + A_j$,
2. initial position $\vec{X}_{m,t} = \frac{\vec{X}_{i,t}+\vec{X}_{j,t}}{2}$,
3. desired density $D_m = D_i$,
4. desired velocity $S_m = S_i$, and
5. goal $\vec{G_m} = \vec{G_i}$.

As in the split situation, the method computes R_m, $M_{m,t}$ and finally $\vec{V}_{m,t}$. Again, legion m cannot split during 30 frames as defined in Section 4.1.2. During such time, the legion m can keep merging again and successively, until there is not other legion that satisfies the conditions to merge.

5. LODUS

Population mobility [4] studies the large scale flow of a certain population in a given geographic space, not necessarily concerning itself with the accurate description of singular individuals but the population as a whole. This can have a varying degree of granularity, a population can be studied in a given city, country, continent or even the entire world. The same is valid on the temporal axis, data can be daily, season, yearly and so on.

To capture these characteristics in a population simulation, the Virtual Humans Lab (VH-LAB) developed LODUS, a framework designed to assist the creation of simulations with varying degrees temporal, physical and population scopes. LODUS is composed of two main modules, the population mobility simulation and a level-of-detail environment generation tool. The level-of-detail environment generation tool module was developed by Gabriel Fonseca Silva as part of his Msc. dissertation. The present works focuses on the population mobility simulation of the framework. LODUS is designed to be used with real world data, and adaptability in mind when generating the routines and population behavior. Figure 5.1 shows an overview of entire LODUS framework, population mobility and city generation modules, and its relationships with a data driven simulation design, the red encircled area highlights the framework sections covered in this work.

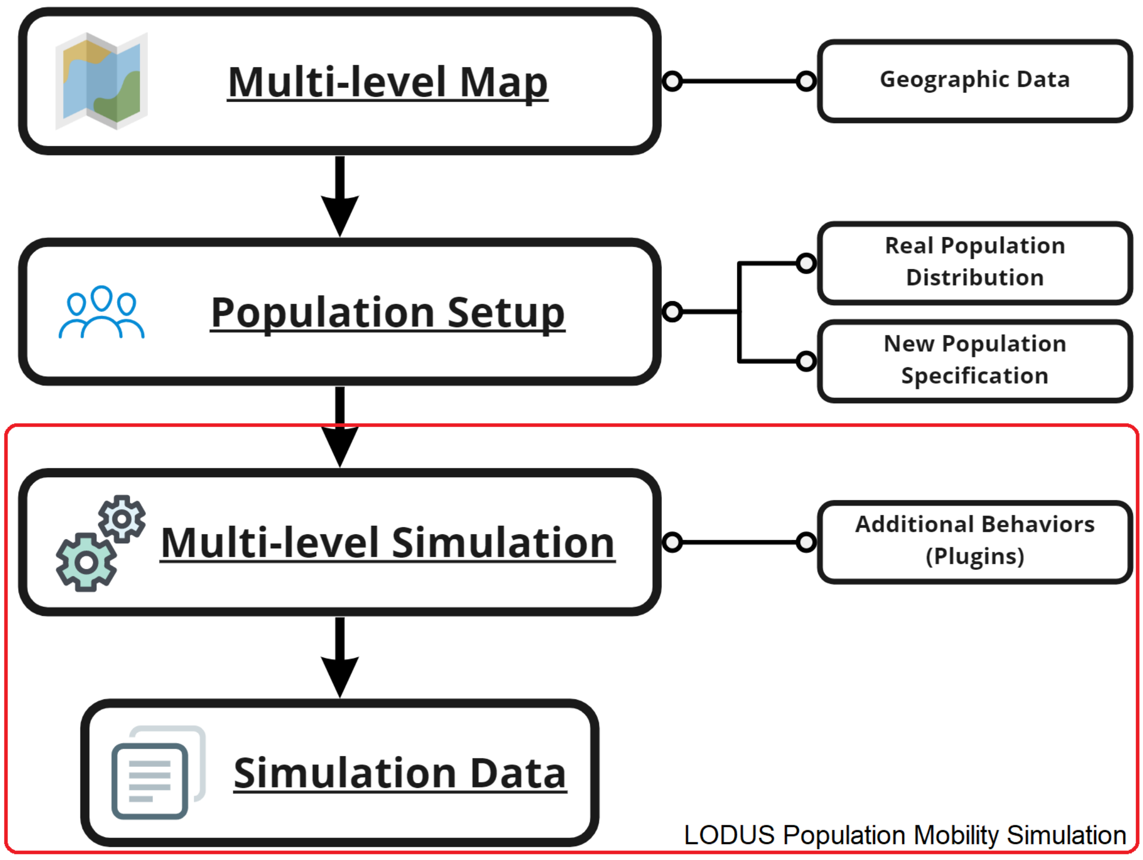

Multi-level Map
Geographic Data
Population Setup
Real Population Distribution
New Population Specification
Multi-level Simulation
Additional Behaviors (Plugins)
Simulation Data
LODUS Population Mobility Simulation

Figure 5.1: Overview of the LODUS framework. Showing the dataflow the framework proposes, from data to city generation and population simulation. The highlighted region marks the modules covered in this section.

Simulation models usually are specific on the set of data they use to perform a simulation which, due to the unreliability of data in the field of population mobility often missing, can be incomplete or otherwise unavailable. We design LODUS as a tool to aid in the configuration of simulators. As such, LODUS does not have a specific set of data to describe a population and its mobility patterns, but it presents a possibility to model a simulator according to the available data. In this manner, a simulation designer can describe a population based on what data is available. In a similar manner, the behavior of a population can also be tailored to what data is available. Next sections contain a top-level description of the framework structures for population, environment and mobility behavior.

Crowd simulation describes how a population moves and evolves on their environment over time. Traditionally, crowd simulation focuses on the modeling of small crowds (on the context of entire populations) [22]. Macroscopic crowd simulations are similar in the sense that individuality they can be abstracted, but population mobility in general deal with larger scopes, e.g. cities, countries and so on. As such we aim to bring crowd simulation strategies to human mobility simulations.

Inspired by Gonzalez et al. [8] whose concluded that humans generally go back to similar locations regularly, and using the abstract representation proposed in BioClouds [2] and Legion [1] models, we propose modeling the population of a simulation as blobs of people. We called blob as an abstract structure, like clouds or legions, but applied to more crowded groups that exist in high level hierarchy of environment, for instance in cities or neighbors. Contrary to BioClouds and Legion, the blobs contains individual and groups characteristics, and evolve in an environment that contains locations which can request specific groups of population. It characterizes an inversion of behavior responsibility, making the environment responsible for decision taking regarding the population movement during the simulation, i.e., locations asks for people, instead of people choosing where to go. For example, a workplace routinely requests workers during morning and homes request workers back, after their shift. This leads to the idea that places have routines that move people in a city. Expanding on the idea of clouds of people, we can add a characteristic profile to each one of these blobs or groups. This does not add to the simulation the idea of a singular separable individual, but does describe a population as having a certain profile that distributes the population differently in temporally and spatially, e.g. children going to the school in the morning and coming home in the afternoon, or workers going to work in the morning but only coming back home at night.

A LODUS population mobility simulation is composed of the following elements: a) The population modeled as blobs which can split and merge. b) An environment modeled as a graph of regions and points of interest which blobs can occupy. And c) a set of population movement actions and routines, to describe the desired population behavior during the simulation.

5.1 Population

To describe populations in LODUS we expand on ideas proposed in BioClouds and Legion, the abstraction of people as groups which can split and merge. These blobs of people should hold characteristics as a group, and not track individual characteristics. When modeling an abstracted group of individuals, characteristics can be modeled as a set of histograms, each describing the distribution of a single characteristic in a given population blob, i.e. a histogram to model age, with possible bins *child*, *adult* and *elder*, will have a certain quantity attributed to each bin, representing that population. In this manner, we can model population as a collection of characteristics, e.g., age, level of schooling, income, etc. And if we want to split and merge different populations we can do so by operating directly of those histograms. This approach presents an information tracking issue if some characteristic of a population is to be associated with a specific subgroup of that population, e.g., which percent of the elder population is vaccinated. To provide the possibility for this association we organize characteristics into population profiles, i.e. blobs contain profiles of characteristic histograms, for instance the vaccinated population contain 10 *children* and 2 *adults* and the unvaccinated population contains 5 *adults*.

A blob i is a tuple (F_i, P_i), where F_i is a set of profiles, e.g., $infected$ or $vaccinated$. Then, F_i is also a tuple $(N_{i,f}, C_fi, f, L_{i,f})$, where $L_{i,f}$ is the population size (scalar value) in this profile, $C_{i,f}$ is the set of the characteristic histograms for that blob and N_f is an identifier (can be a string) and states for the characteristic this profile represents, e.g. $infected$. A blob with a quantity p of individuals will have p individuals distributed between all of its profiles, e.g. a blob with population=10 can be distributed as follows: 5 vaccinated and 5 unvaccinated. The set C_fi, f contains histograms, where a histogram h is a tuple (K, B), associating characteristic K to bins B, e.g., *age*, with bins 30 *adults* and 20 *children*, and *income*, with bins 20 *high* and 20 *low*. Each histogram h in a profile f contains population in its bins equal to $L_{i,f}$. Still in the blob i, population P_i will be the sum of populations at each profile contained in F_i. Figure 5.4 contains a diagrams of a blob, showing its profiles and contained histograms.

Blobs can be operated in 3 ways: blobs can split, can be merged or have some quantity of population from one of its profiles moved to another, for example, moving population from the $susceptible$ profile to $infected$. These operations can use population templates as parameters.

A blob split is an operation which separates a blob into two smaller blobs. One way to split a blob in two can be the simple division of all characteristic histograms between the newly created blobs. For example, splitting the blob b with p individuals, with produce blobs c and d, with q and r individuals respectively, where $p = q + r$, this is valid for every blob profile. The splits however do not have to distribute equally each profile, e.g., when creating blobs i and j in a split, it is possible that blob i obtains all $children$ from an original blob and blob j ends with 0 $children$. This behavior lets us separate groups of population with particular characteristics, for example, a group of children that will be sent to school, or a group of workers that will be sent to work. The specification of affected population is done with population templates, if no specification is used,

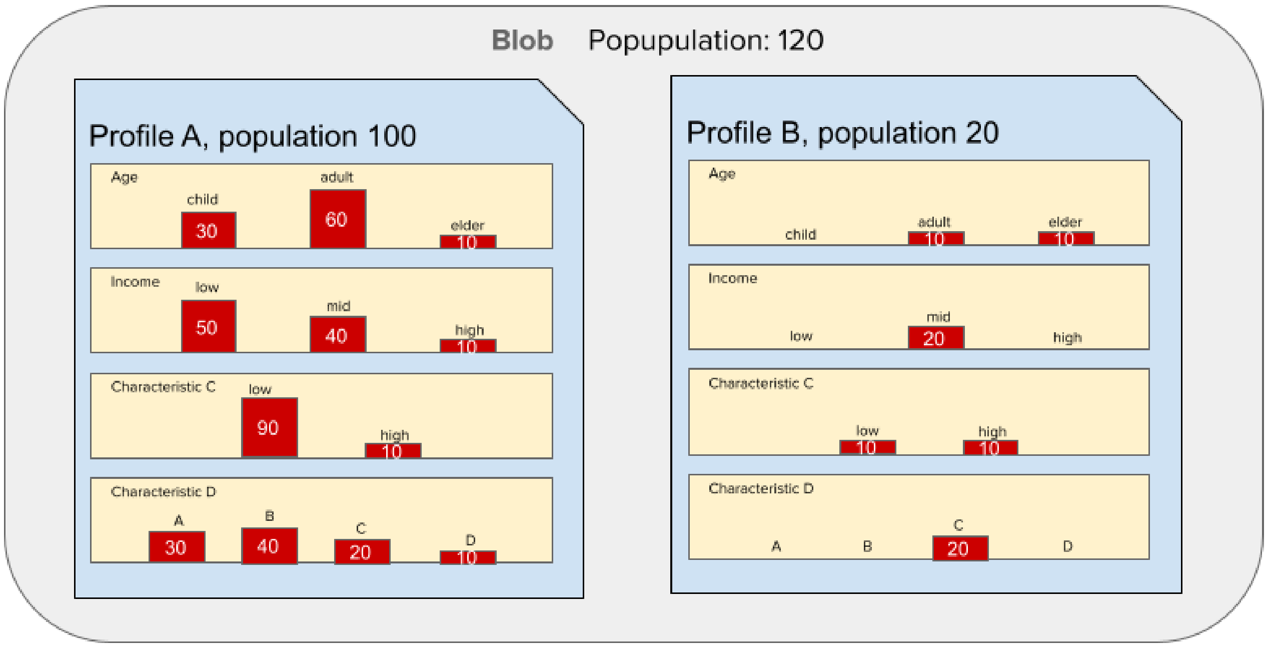


Figure 5.2: Blob diagram showing a blob containing two characteristic profiles. In this example each characteristic profile contains four characteristic histograms. Note that each histogram in a given profile contains population equal to that profile population. Also note that the blob population is the sum of populations from each profile.

population is split weighted according to population at each bin. Figure 5.3 shows a diagram of a blob split.

Conversely, a blob merge operation is an operation which merges two separate blobs into a larger blob. Adding blobs c and d, with q and r individuals respectively, it will produce a new blob with p individuals, where $p = q + r$. This addition process occurs per population profile for the blobs. This operation preserves population quantity, both in an global level and in a profile level. That is, the total population does not change in the simulator, and the total population for each profile does not change. Figure 5.3 shows a diagram of a blob merge.

The third blob operation aims to move population from a profile a to a profile b inside a blob. This population transfer does not change the total population in a blob, but changes its characteristics. A population profile transfer can simulate situations like vaccination, infection and so on. The main advantage of this, is tracking a population profile which acquired a certain characteristic, e.g., what is the percentage of a population which is vaccinated among different groups.

Population templates are a set of mappings that describe which population is affected by any given operation. A template T affecting a blob i describes a set of pairs (k, v), where k is a characteristic in C_i for blob i, and v is one of the possible values for a bin in the associated histogram. A characteristic which is in C_i but not defined in T is assumed to be affected according to its population distribution, i.e., weighted by population. The template T can contain any number of pairs defining a population to be affected. For example

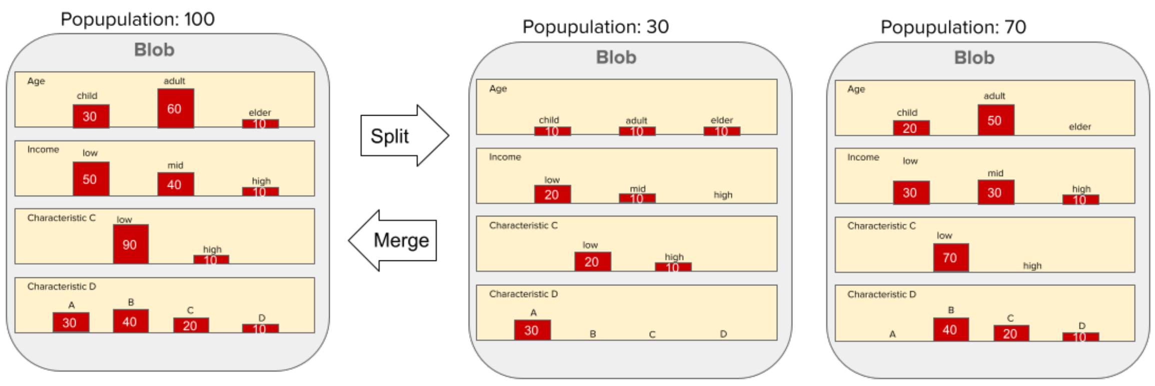


Figure 5.3: Diagram showing split and merge operations. Split separates a blob population in two. Merge groups two blobs into one population. This diagram omits profiles for clarity, but the same behavior applies to each profile.

5.2 Environment

To adequate our simulation framework to the varying scopes of space population mobility, we define the space as an abstract graph representing *regions* and *points-of-interest*. Regions represent a large region of physical space in the world, containing points-of-interest. Points-of-interest (POIs) represent the places that motivate human travel in the simulation. Regions and POIs can represent places or areas of any spatial scope, this definition is done during simulation design by the user. In that manner, a simulation region can represent a country and a POI can represent a state, or in another example, regions can represent neighborhoods of a city and POIs represent the facilities, residential and commercial zones in those neighborhoods.

Regions model a grouping of POIs, which represent an interesting characteristic to model movements over larger spans and to track origin and destination of a given population over a period of time. In this way, a region i is a container for a set of POIs P_i. Regions model the larger level-of-detail for the environment description, e.g., in a city simulation, regions can model neighborhoods.

In turn, POIs model the finest level-of-detail for the environment description. Population blobs are contained in POIs and move between them as the simulation progresses. A POI p is a tuple (B_p, R_p), where B_p is the set of population blobs contained in p, R_p is the routine p follows. This routine describes the demands for population this POI has over time, e.g., a super market requests consumes, a school requests students, and a residential area provides individuals to other regions. As humans generally tend to be cyclical in their behavior, repeating similar routines daily, a LODUS simulation is also cyclical. As such a routine repeats every cycle of simulation. This cycle can represent days, weeks and so on, depending on how a designer creates POI routines.

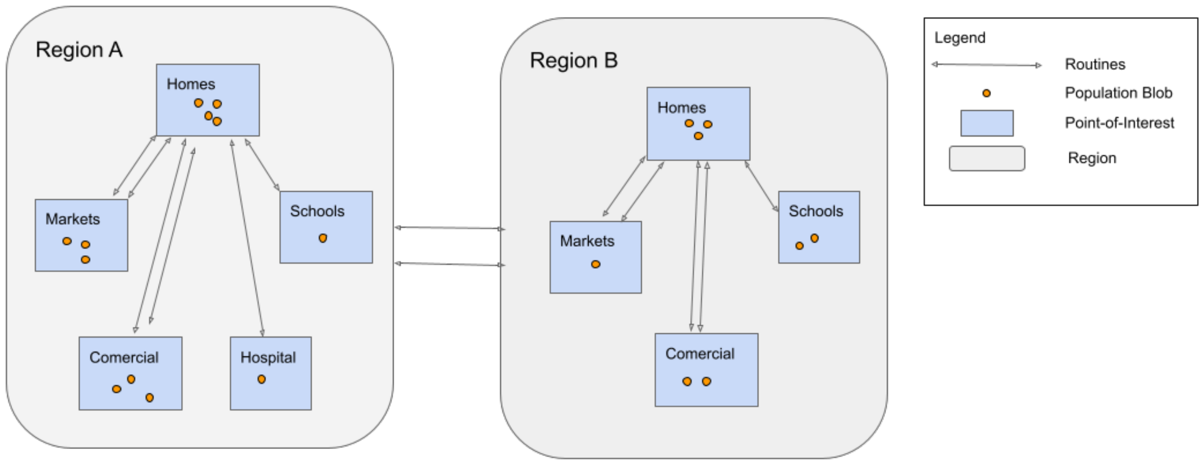


Figure 5.4: A diagram of a LODUS environment.

5.3 Movement Actions and Routines

The movement of blobs during a simulation is defined by the POI routines and other movement actions defined by the simulation designer. A movement action moves a quantity of population between any number of POIs or between profiles in a blob, in a simulation step. We designed LODUS to be extensible, and as such, its set of available actions is extensible, a user can add new movement actions to a simulator. New actions can be used to model different life situations, for example, bus transportation, events or crisis situations.

The system of movement actions is defined in two parts, the action description and the function which consumes that description. Action description is a pair of action type, a identifier for this action, and the set of values this action expects to receive as parameters. The definition of values to be contained in the parameter set is contractually defined by the simulation designer. Actions can be either base actions or complex actions.

The base movement action type is *move population*, to operate the simplest possible simulation operation, moving a quantity of population from one POI to another. A *move profile* action is the base action that changes the population profile for a blob, moving population from one profile to another. Base actions are actions which directly operate change in the simulation.

Complex actions, on the other hand, can be broken down into a series of base actions, describing complex mobility behavior. An example of a complex action is returning all children from schools to their respective homes, this action is broken down into several *move population* actions, moving population from a school POI to their original home POI. As such, complex actions return a list of sub-actions and base actions return empty lists.

Actions accept one special parameter in their parameter dictionary, a population template. As described in Section 5.1, a population template is a set of key-value pairs, describing which population characteristic histograms to affect. For example, a population template mapping of $(age, adult)$ will cause the defined movement action to only affect the population in the $adult$ bin of the age characteristic histogram. Undefined pairings are assumed to be affected uniformly weighted by the contained population. The *move population* action operates this way by default.

The *move population* base action has arguments (Po, Pd, Q, T) where Po is an origin POI, Pd is a destination POI, Q is a desired quantity to be moved and T is a population template. This action will split a blob with population Q from the blobs occupying Po and merge the split blob to Pd while respecting the mappings present in T. In case there $size(Po, T) < Q$, *move population* moves as much population as possible, where $size(n, t)$ is the quantity of population in POI n which respects template t. The *move profile* base action has arguments (B, Ko, Kd, Q, T) where B is a target blob, Ko is the origin profile, Kd is the destination profile, Q is a desired quantity to be moved and T is a population template. The action will subtract Q population from profile Ko and add it to profile Nd while respecting the mappings present in T. In case there $size(Ko, T) < Q$, *move population* moves as much population as possible, where $size(K, T)$ is the quantity of population in profile K which respects template T.

The extensibility of LODUS comes in the forms of plugins which allows a designer to add more types of actions. A plugin adds to the simulator another function associated to an action type. Plugins extend the set of actions available for the designer during a simulation description. A plugin, is a extension to the simulator, which provides an association of a movement action type and a function with a set of contractually defined parameters. Added movement actions can be both base or complex actions.

The inversion of responsibility for movement in LODUS puts the movement behavior of the simulated population on POIs instead of the population. Designers can achieve the movement behavior in four different manners: routines, global routines, direct calling of movement actions and queuing actions for future frames. Table 5.1 shows a relation of each type of action call and their proposed uses.

Simulation Routines model the repeating temporal behavior of any given POIs, with what parameters and during which simulation steps each movement action should be called. These routines are cyclical, repeating every d simulation steps, where d is some user defined value to represent the duration of a cycle. Each in the cycle step models the smallest time increment desired. For example, a d of 24 can represent a cycle with duration of one day with each step modeling an hour, or a cycle of 24 days with each step modeling a day. The semantics for a step and cycle duration is defined by the mobility functions the designed simulation uses.

A routine i is modeled as a list of pairs $(C, A_{t,i})$, where C is the is the *cycle moment* when action $A_{t,i}$ is called. The *cycle moment* C for simulation frame Fr is calculated as $Fr \mod d$, and

Type of Action Call	Frequency	Example of Use
Routine	Updates according to each POI's routine.	Modeling regular city cyclical behavior and human mobility.
Global Routine	Repeats every n frames.	Modeling global events, like random-walks and infection spreading.
Next Frame Queue	Adds an action to be called once next frame.	Models actions which resolve their effects over multiple simulation steps.
Direct	Calls an action once.	Allows for user input during a simulation. Allows for one time events during specific frames.

Table 5.1: Table showing different ways to call actions in a LODUS simulation. Each type of action call is better suited for some specific behaviors.

as such, action $A_{t,i}$ is called when $C = Fr \mod d$. This repetition of a population's behavior was inspired by Gonzalez et al. [8], which model proposed the real life activities in real-life, tracked by geo-tracking cellphones.

Global Routines(GR) model a similar behavior as routines, except they apply to every POI in the simulation. A GR is a pair (C, A) where C is the *cycle moment* when action A is called for all POIs. This behavior could be described by adding an extra action to each POIs routine and is here for ease of use by designers of a simulation.

Direct call of actions allows designers to program specific one-time events in the simulation. This invocation is done by simply feeding the simulator a new action after any given step of a simulation. For example, modeling a cultural event to happen during a specific simulation frame Fr, can be achieved by processing the action modeling that event, one time, after frame Fr is processed.

Queuing actions for future frames can be done in the same manner as direct invocation of actions or by actions processed in routines. This lets actions do spillover for future frames, for example, population movement where there is no available public transportation during a given time can prolong movement through several frames.

5.4 Simulation Algorithm

During a simulation, LODUS executes the following algorithm:

Algorithm 1: LODUS population mobility simulation flow

```
$B \leftarrow$ base action set;
$D \leftarrow$ routine cycle duration *steps* $\leftarrow$ simulation step count;
$G \leftarrow$ load simulation environment;
$i \leftarrow 0$;
// Add all required action functions to the environment.
G.initialize_plugins();
while steps $> i$ do
    // The action queue for this frame.
    Queue $Q$;
    if $i < 0$ then
        // If there is any actions leftover from last frame.
        $L \leftarrow$ frame $i$ - 1 queued actions;
        $Q.queue(L)$;
    end
    // Current frame routines. That is, the routines for a i $\mod$
       D step of the simulation.
    $F \leftarrow$ frame $i \mod D$ routines;
    $Q.queue(F)$;
    // Current frame Global Routines.
    $R \leftarrow$ frame $i$ global routines;
    $Q.queue(R)$;
    // While there are still actions to consume this frame.
    while $Q \neq \emptyset$ do
        // Remove action $q$ from queue.
        $q \leftarrow Q.dequeue()$;
        // Consume action $q$. If $q$ is a complex action $g$ receives a
           new list of actions. Otherwise $g$ receives an empty list.
        $g \leftarrow G.consume(q, i)$;
        // If $q$ is a complex action add its sub actions to the queue.
        if $q \notin B$ then
            $Q.queue(g)$;
        end
    end
    // Consume any user defined action for this frame.
    $u \leftarrow$ user defined action (direct action call);
    $G.consume(u, i)$;
    $i \leftarrow i + 1$;
end
```

Where Q is queue which accepts lists as arguments, $G.consume(q, i)$ is the function which consumes an action q and effects the changes defined in the contract for action q. In case q is not a base action, it returns a list of new actions to be queued up. Action u is any number of optional user defined actions to be invoked at any frame during simulation, it is part of the algorithm to explicit it is not part of the regular routine/global routine and queued actions simulation flow.

5.5 Data Input and Implementation Details

The setup for a LODUS population mobility simulation requires 3 parts: a simulator script, an environment and population description input and the action plugins. The simulator script is a python script implementing the *EnvGraph* simulator class. The environment and population descriptions are defined in a *JSON* file. Finally, plugins are python classes used in the simulator script.

Data Input File Description

The environment and population and description file is a JSON object with the top level structure shown in Listing 5.1:

```
{
    "population_template" : <population histogram description>,
    "block_types" : <population profiles>,
    "repeating_global_actions" : [ <global routines> ],
    "regions" : [ <region descriptions> ]
}
```

Listing 5.1: Top level structure for the environment and population description file

The characteristic histograms each population blob has are described in the *"population_template"* key-value pair. *<population histogram description>* is a JSON object containing a key-value pair for each characteristic histogram desired in a simulation, each value in the pair is a list with the possible bins for that characteristic. Listing 5.2 shows a characteristic histogram example:

```
"population_template":
{
    "age": [
      "adults",
      "elders",
      "young"
    ],
    "occupation": [
      "idle",
      "student",
        "worker"
    ]
}
```

Listing 5.2: Population characteristic histogram description.

Population profiles are defined in the *"block_types"* key-value pair. <population profiles> is a list of all desired profiles in the simulation. Listing 5.3 shows a profile description example:

```
"block_types": [ "healthy", "infected" ]
```

Listing 5.3: Populaiton characteristic profile description.

The environment is described in the key-value pair *"regions"*. The content of *"regions"* is a list of JSON objects which follow the pattern in Listing 5.4:

```
"regions":
[
    {
    "name": "Aberta dos Morros",
        "image_position": [ 1150.0, 3022.0 ],
        "long_lat_position": [ -51.1975768802765, -30.1564777842068 ],
        "world_position": [ 2212.175202540122, -15068.325869652443 ],
        "population": [ 7146, 0 ],
        "population_description": {
        "healthy":
        {
          "age": { "adults": 3574, "elders": 1429 },
          "occupation": { "idle": 2143 }
        },
        "nodes": { <POI description> }
      },

    }
]
```

Listing 5.4: Population characteristic histogram description.

Where *"image_position"*, *"long_lat_position"*, *"world_position"* are optional location parameters for this region, in image position, geographic and world positions respectively. The *"population"* key is a list containing the quantity of population contained in each population profile for this region and *population_description* is a distribution of that population over the population profiles and their respective bins. Missing values and keys for population description assumes the remaining unassigned population is supposed to be distributed uniformly. The initialized population is assigned to the *"home"* POI by default.

Points-of-interest (POIs) are described in the *"nodes"* key-value pair containing a JSON object where each POI is mapped as a key. The content of each described POI is a JSON object with a key *"characteristics"* containing any information the user desires to use in plugins relative to this POI, and any quantity of number keys. Each of the number keys defines an hour of the routine for this POI, and contains a list of possible actions. Actions themselves are require two keys: *"type"* a plain text identifier for this action, according to its plugin, and *"values"* the contract defined parameters for this action. Listing 5.5 shows and example of a defined POI containing two actions and a position characteristic:

```
"home":
{
    "characteristics":
    {
        "world_position": [ 1694.9063426768407, −15259.419436546974 ]
        },
    "7":
        [
        {
          "name": "push_adults_to_work",
          "type": "push_population",
          "values": {
            "origin_region": "Aberta dos Morros",
            "origin_node": "home",
            "destination_node": "work",
            "quantity": 5716,
            "population_template": {
              "age": "adults",
              "occupation": "worker"
            } } }
        ],
    "18":
        [
        {
          "name": "return_workers",
          "type": "return_population_home",
          "values": {
            "region": "Aberta dos Morros",
            "node": "home",
            "quantity": −1,
            "population_template": {
            } } }
        ]
}
```

Listing 5.5: POI description.

Global routines are described in the key *"repeating_global_actions"* key, as described in Listing 5.1. Global routine descriptions follow the same structure as regular routines with two exceptions, the keys *"node"* and *"region"* in the *"values"* object can be omitted and global routines have an extra *"cycle_length"* parameter. Listing 5.6 shows an example of global routine description.

```
"repeating_global_actions": [
    {
        "cycle_length": 1,
        "type": "levy_walk",
        "values": { "population_template": {} }
    }
]
```

Listing 5.6: Global Routine description.

Plugins describe additional actions and hold the external parameters the simulation designer desires for the added functions. This provides a way to customize the simulator to model different types of human mobility situations. Plugins are written as python modules to be imported by the simulator script. Each module should describe a class inheriting the LODUS TimeActionPlugin class, this makes them able to be loaded by the LODUS environment. Listing 5.7 and Listing 5.8 shows an example of a plugin which adds an example action to the simulator. The example action moves 100 population from a POI a at region r to the POI $home$ at region r.

```
class ExamplePlugin(environment.TimeActionPlugin):

    def __init__(self, env_graph):
        super().__init__()

        self.graph = env_graph
        self.set_pair('example_action', self.example_action)

        self.example_parameter = 'foo'
```

Listing 5.7: Plugin initialization.

```
    def example_action(self, values, hour, time):
        region = values['region']
        if isinstance(region, str):
            region = self.graph.get_region_by_name(region)

        node = values['node']
        if isinstance(node, str):
            node = region.get_node_by_name(node)

        pop_template = values['population_template']

        new_action_values = {}
        new_action_type = 'move_population'
        new_action_values['origin_region'] = region.name
        new_action_values['origin_node'] = node.name
        new_action_values['destination_region'] = region.name
        new_action_values['destination_node'] = 'home'

        new_action_values['quantity'] = 100
        new_action_values['population_template'] = pop_template

        action = environment.TimeAction(_type = new_action_type,
                                        _values = new_action_values)

        return [action]
```

Listing 5.8: Example plugin action function definition.

The simulator script has three functions: initialize the environment, load plugins and control the simulation flow. Initializing the simulation environment uses the data input file described above and the framework class *EnvGraph*. Loading plugins is done by initializing the plugin classes and loading the objects into the EnvGraph. Controlling the simulation flow is done by a loop controlling the step update for the simulator. Listing 5.9 has a simulator script example:

```
# Data Loading
env_graph = generate_EnvironmentGraph(data_input_file_path)

# Parameters
days = 3
day_duration = 24
env_graph.routine_day_length = day_duration
simulation_steps = days * day_duration

# Load Plugins Examples
plug = ExamplePlugin(env_graph)
plug.example_parameter = 'bar'
env_graph.LoadPlugin(plug)

# SIMULATION
for i in range(simulation_steps):
    # Routine/Repeating Global Action call example
    # Updates POI Routines and Global Routines
    # These are defined in the input environment descriptor file
    env_graph.update_time_step(i % day_duration, i)

    # Direct Action Call example for frame 50
    # if i == 50:
    #     dummy_action = TimeAction('push_population',
    #                               {'region':'example1',
    #                               'node':'example2',
    #                               'quantity':50})
    #     env_graph.direct_action_invoke(dummy_action)

    # Next frame queue action example for frame 60
    # if i == 60:
    #     dummy_action = TimeAction('push_population',
    #                               {'region':'example1',
                                    'node':'example2',
                                    'quantity':50})
#       env_graph.queue_next_frame_action(dummy_action)
```

Listing 5.9: Simulation controller script example.

5.6 LODUS Case-Studies

To serve as case studies for LODUS, we modeled three possible mobility situations: urban mobility, public transportation and disease spreading. Of these case studies, we focused our implementation efforts on the urban mobility models, and present theoretically the possible models for public transportation and disease spreading. Section 6 contains experimental results for our urban mobility simulations.

5.6.1 Urban Mobility

Gonzalez et al. [8] makes two useful affirmations on the movement of urban populations we can use as a starting point for our abstracted urban mobility model. Notably, the conclusion that humans repeatedly travel to a few choice locations and that there is some degree of noise to their daily movement which can be described with a Lévy flight random walk. A Lévy flight is a random walk pattern which follows a Lévy probability distribution for the length of each jump. A Lévy distribution is a heavy-tailed probability distribution. In practice, this leads the Lévy flight to have a high concentration of short jumps and a lower concentration of longer jumps. The parameters of the Lévy flight change depending on the population being measured. With these two assumptions, we instantiated two different behaviors: a rigid daily routine where people move into their favorite POIs, and a Lévy flight noisy movement which moves people to random POIs.

The daily routine distributes the *"adult worker"* population of a *"home"* POI to every *"work"* POI, weighted by distance. The action also takes a scaling parameter, to model variation of mobility over time, e.g. weekends have less traffic of people than work days. This mobility description is deterministic when measuring quantity of people, but not the composition of the moved blobs, that is, characteristics other than age and occupation varies from frame to frame, as blobs split and merge. The noisy Lévy flight population movement, developed in LODUS, is a global routine which samples a distance using a Lévy distribution and moves some quantity of population to a POI that matches that distance. For each POI e compute a histogram of distances to other POIs, which are binned by a distance grouping value. When called, the action divides the population of a POIs in packets to be moved, each packet has a chance to be moved, defined by a mobility scale parameter. When a POI selects a distance, it will send population to one of the POIs in that distance group, randomly chosen. Distance values with no POIs will cause the populations to stay in the origin POI.

This Lévy plugin has a scale parameter s which controls the Lévy distribution. A mobility scale parameter l controls how often the population moves and a packet size parameter p, which models the size of groups which move through random walk. Section 6 presents experiments using this model. The experiments compare different values for each parameter, and different sizes of city. Table 6.4 shows the experiments performed.

5.6.2 Public Transportation

One of the main concerns of public transportation scaling is making sure the traffic system is capable of handling the required flow of population during the day. People traveling during rush hours can clog traffic in a way where the travel time becomes increasingly abnormal in relation to normal traffic. Modeling this rush hour delay lost in traffic can be modeled as actions which take several steps to finish taking effect on the simulation.

We can add the delay to our model by creating the *delayed move population* action to have a route parameter R, to decide which route going from POI *a* to POI *b* will take, and the maximum capacity and flow rate for R. This can be done with a map to store the transportation parameters, and a limitation to the *move population* action. Movement which would overflow the transportation restrictions should be queued for future frames, as shown in Table 5.1.

The way this *delayed move population* action should follow the algorithm, where $D(Po, Pd, quant)$ is a *delayed move population* action requesting the move of $quant$ population from POI Po to POI Pd. And, $M(Po, Pd, quant)$ is a *move population* action requesting the move of $quant$ population from POI Po to POI Pd. $Route$ and $route_capacity$ are functions which retrieve the route properties between two POIs:

Algorithm 2: Delayed *move population* action pseudocode.

```
$Q \leftarrow$ next frame queue;
$quant \leftarrow$ quantity to be moved;
$Po \leftarrow$ origin POI;
$Pd \leftarrow$ destination POI;
$R \leftarrow route(Po, Pd)$;
$C \leftarrow route\_capacity(R)$;
// If requested movement overwhelms route.
if $quant > C$ then
    // Queue remaining quantity for next frame.
    $N \leftarrow D(Po, Pd, quant - C)$;
    $Q.queue(N)$;
    // Return a movement action at capacity.
    return $M(Po, Pd, C)$
else
    // Return a movement action for $quant$.
    return $M(Po, Pd, quant)$
end
```

5.6.3 Disease Spreading

Recently, disease spreading models are of increased interest due to the COVID-19 epidemics. While several models abstract away the spatial factor, we can combine them with LODUS to try and add those mobility factors back to the simulation. We propose using a global routine in LODUS to compute localized *SIR* [9] infection simulations for each POI. While *SIR* simulations compute infections over the entire population, we propose using LODUS as a population mobility based restriction to infections. This simulated population will possess, as described in Section 5.1, to represent the characteristics of a population in a *SIR* simulation: *susceptible*, *infected* and *removed*.

The interaction of an external infection and disease spread solver and the LODUS framework is possible with a plugin which manages the infection procedure. To that end, we propose modeling a simplified *SIR* simulation which operates on a target group of the total population. This *SIR* simulation has γ and β parameters defined by the user. As such, we model a global routine, as shown in Table 5.1, to perform several *SIR* steps infections over the environment, each targeting a specific POI of the simulation. The *SIR* infection applied to a POI, affects only the population present in that POI. The parameters S, I and R for the *SIR* simulation, are taken from the sizes of the population with each profile in the POI. This lets the *SIR* simulation always compute the 'current' state of infection for any POI, even if population moves to other places. Subsequently, the population in the affected POI will move to other POIs. This has the effect of restricting contagion to already infected places and slowly diffusing the infection over the environment.

6. EXPERIMENTAL RESULTS

This section shows our experiments for BioClouds, Legion and LODUS. Section 6.1 contains the experiments for BioClouds model. Section 6.2 contains the experiments for Legion. And Section 6.3 contains the experiments for LODUS, simulating the city of Porto Alegre under various conditions and different simulation approaches.

6.1 BioClouds Experiments

This section presents some experimental results obtained by our model. Section 6.1.1 shows that the proposed clouds are accurate representation of the agents they are trying to personify, while Section 6.1.2 shows some performance comparison among BioClouds, BioCrowds and other works in literature.

6.1.1 BioClouds Accuracy

The goal of this section is to show that the clouds, in our method, accurately represent the agents inserted on them. To do so, some tests are performed with both BioClouds and BioCrowds, having their respective results compared. We designed three experiments (*A*, *B*, and *C*) in order to compare the average densities and speeds between BioCrowds and BioClouds. Experiment *A* simulates a single crowd of agents, while experiments *B* and *C* simulate crowds of various sizes in a bidirectional flow.

The BioCrowds agents density measurement was performed by dividing the number of agents in a certain cloud by the convex-hull area the cloud occupies. This measurement is not accurate for very small areas and few agents, when densities are unrealistically high. For all three experiments, the MVW (Micrososcopic View Window) is positioned at the center of the environment, with $w = 80$ and $h = 50$. Biocrowds agents are spawned when the cloud 1 enters the MVW and are removed when the agents leave the MVW. The environment in these experiments is a flat square of terrain with no obstacles, measuring 150 by 150 meters.

Experiment *A* simulates one only crowd (1) of agents walking unobstructed. The cloud 1 starts the experiment at a position $(0, 74)$ in a 2D coordinates and moves towards its goal at position $(150, 74)$, at a max preferred speed of $1.3m/s$. Cloud 1 represents 200 agents at a preferred density of 0.5agent$/sqm$. Figures 6.1 and 6.2 show the obtained density and speed results for experiment *A*, respectively. The graphs compare how closely the cloud density and speed (in blue), relate to the density and speed of agents which are spawned (in red). In an unobstructed scenario, agents follow closely the behavior of the cloud which created them, as shown by the low deviation between the red and blue lines. The small variations are due to the microscopic simulation where agents avoid collisions and interact with others, what does not happen in BioClouds. In fact, in the same time we show that both behaviors are coherent, we show the main advantage of our method when allowing to perceive (and quantitatively measure) the impact of individuals in the simulation.

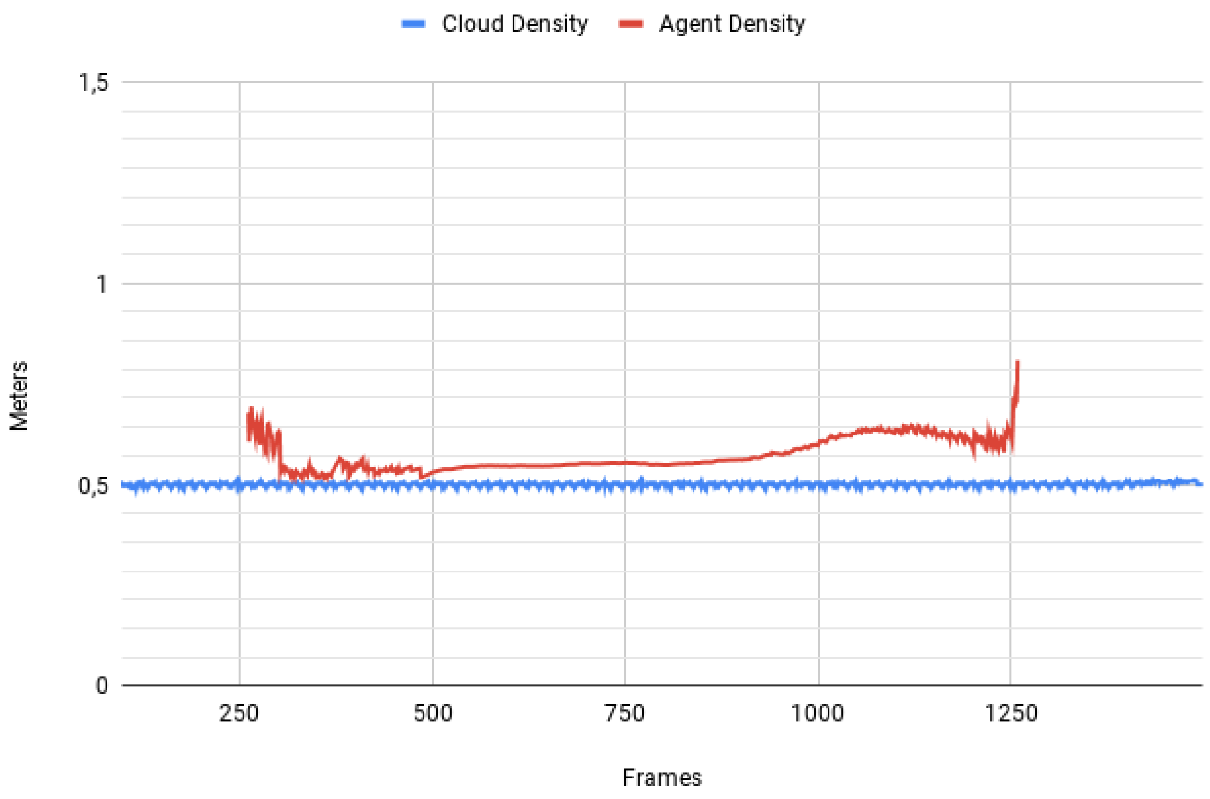


Figure 6.1: The densities for experiment *A*. Cloud density in blue. Agents density in red.

Experiment *B* simulates two equal sized crowds (names 1 and 2) of agents walking towards each other in a bidirectional flow. Cloud 1 starts the experiment at position $(0, 74)$ and moves towards its goal at position $(150, 74)$, at a max preferred speed of $1.3m/s$. Cloud 2 starts the experiment at position $(150, 74)$ and moves towards its goal at position $(0, 74)$, at a max preferred speed of $1.3m/s$. Both clouds 1 and 2 represent 500 agents each at a preferred density of 0.5agent$/sqm$ each. Figures 6.3 and 6.4 show the obtained density and speed results for experiment *B*, respectively. In this experiment it is easy to see the impact of the two crowds when being simulated in a microscopic way while crossing each other. As we can perceive such disturbance caused by the crowds it is not visible in the clouds.

Experiment *C* simulates two different sized crowds of agents (500 and 250 agents) walking towards each other. Cloud 1 starts the experiment at position $(0, 74)$ and moves towards its goal at position $(150, 74)$, at a max preferred speed of $1.3m/s$. Cloud 1 represents 500 agents and has a preferred density of 0.5agent$/sqm$. Cloud 2 starts the experiment at position $(150, 74)$ and moves towards its goal at position $(0, 74)$, at a max preferred speed of $1.3m/s$. Cloud 2 represents 250 agents and a preferred density of 0.5agent$/sqm$ each. Figures 6.5 and 6.6 show the obtained density and speed results for experiment *C*, respectively. Comparing with experiment *B* it is easy to see that

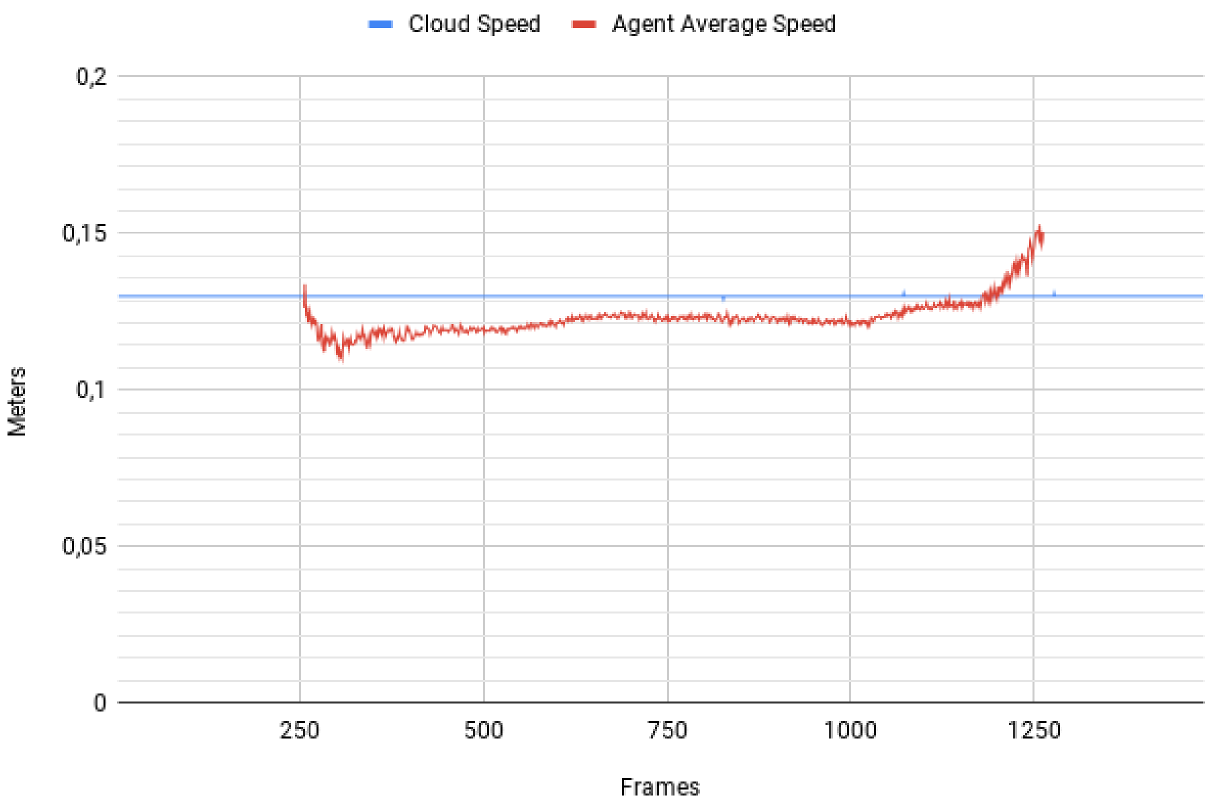


Figure 6.2: The speeds for experiment A. Cloud speed in blue. Agent average density in red.

as there are more agents, higher is a variation due to the number of individuals and their interactions in the simulation. In fact, one can say that our method succeeds to simulate crowds as unities and also to show the impact of individuals in the simulation in the same time.

Figures 6.3,6.4,6.5 and 6.6 show the graphs of distinct crowds interacting. In red and blue the graphs show the values (density and speed) for the clouds. In green and yellow the graphs show the average values for the BioCrowds agents, spawned by our method. It is easy to see that when simulating clouds there is no variation in densities and speeds. On the other hand, the interaction among agents crossing others increases densities above the estimated density by the cloud. As well as reducing speed below the estimated velocity of the cloud. These data indicates that when simulating and seeing clouds there is a expected loss of accuracy what can be observed when simulating our microscopic method. The aspect we want to highlight here is that Clouds or Crowds can be simulated and visualized in specific points, based on user decision, in order to obtain the accuracy, microscopic model provides, and also take benefit from the optimized computational time, allowed by macroscopic model (discussed in Section 6.1.2).

Also, a qualitatively analysis can be performed. Figure 6.7 presents some snapshots of the Experiment B. Two clouds are walking towards their respective goals, placed in opposite directions (Figure 6.7(a)). When the clouds enter inside the MVW area, agents begin to be spawned (Figure 6.7(b)). It is possible to see that such agents are distributed along the area of the cloud

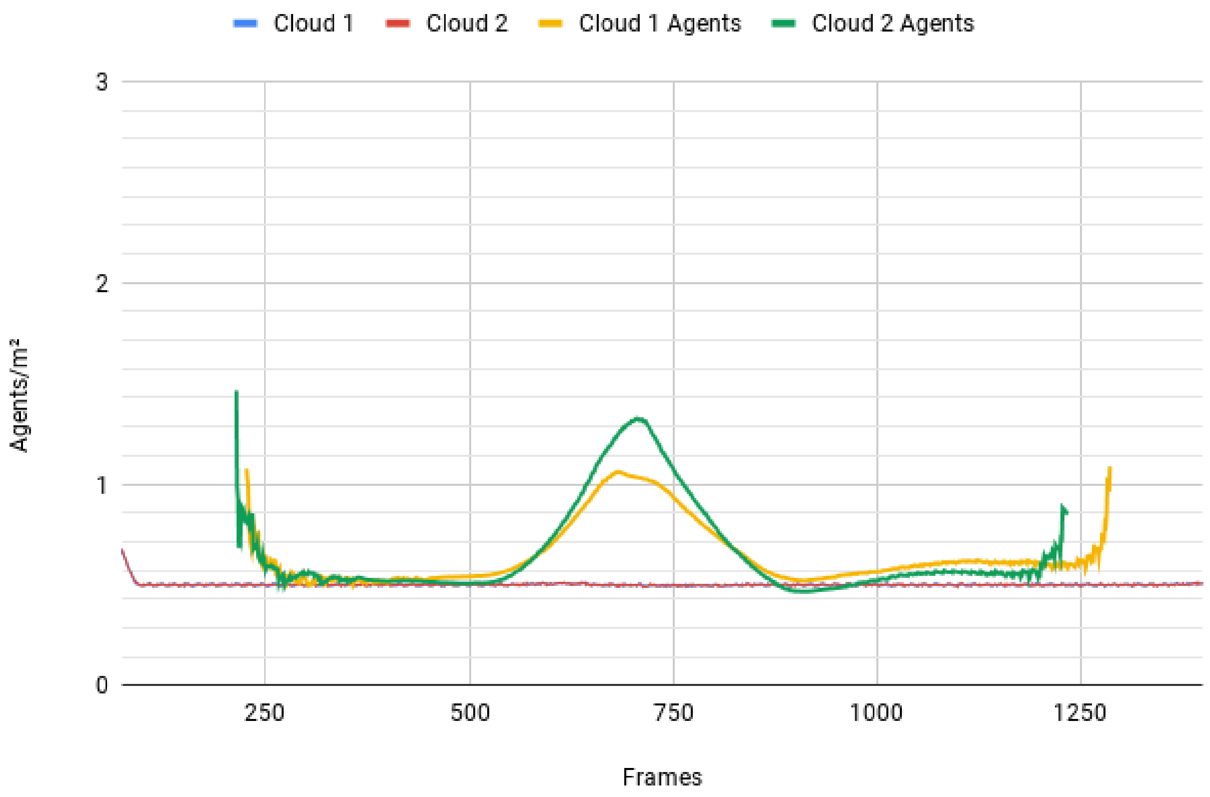


Figure 6.3: The densities for experiment *B*. Cloud densities in red and blue. Agent densities in green and yellow.

which is already inside the MVW. Local interactions between agents can be seen in Figure 6.7(c), where agents of both clouds are forming lanes while crossing each other.

In Figure 6.7(d), it is possible to see the variation the crowd shape adopts in relation to the expected cloud shape. It happens due to local interactions that are simulated in microscopic level. Even so, agents were able to stay nearby their respective cloud center, so, when they start to leave the MVW area, the clouds begin to resurface (Figure 6.7(e)). Therefore, when agents already left the MVW area, only the clouds continue to be simulated. Figure 6.8 shows a zoomed in instance of figure 6.7(c), where the formed lanes are represented with the grey (lanes moving left) and white (lanes moving right) lines.

6.1.2 Performance

The goal of this section is to show that BioClouds is able to achieve better performance when compared with BioCrowds. Also, we evaluate our performance in comparison with the work of Narain et al. [16].

BioClouds performance is dependant on the current radius size of each cloud, since each cloud attempts to capture each cell inside its radius what is used to compute the cloud motion.

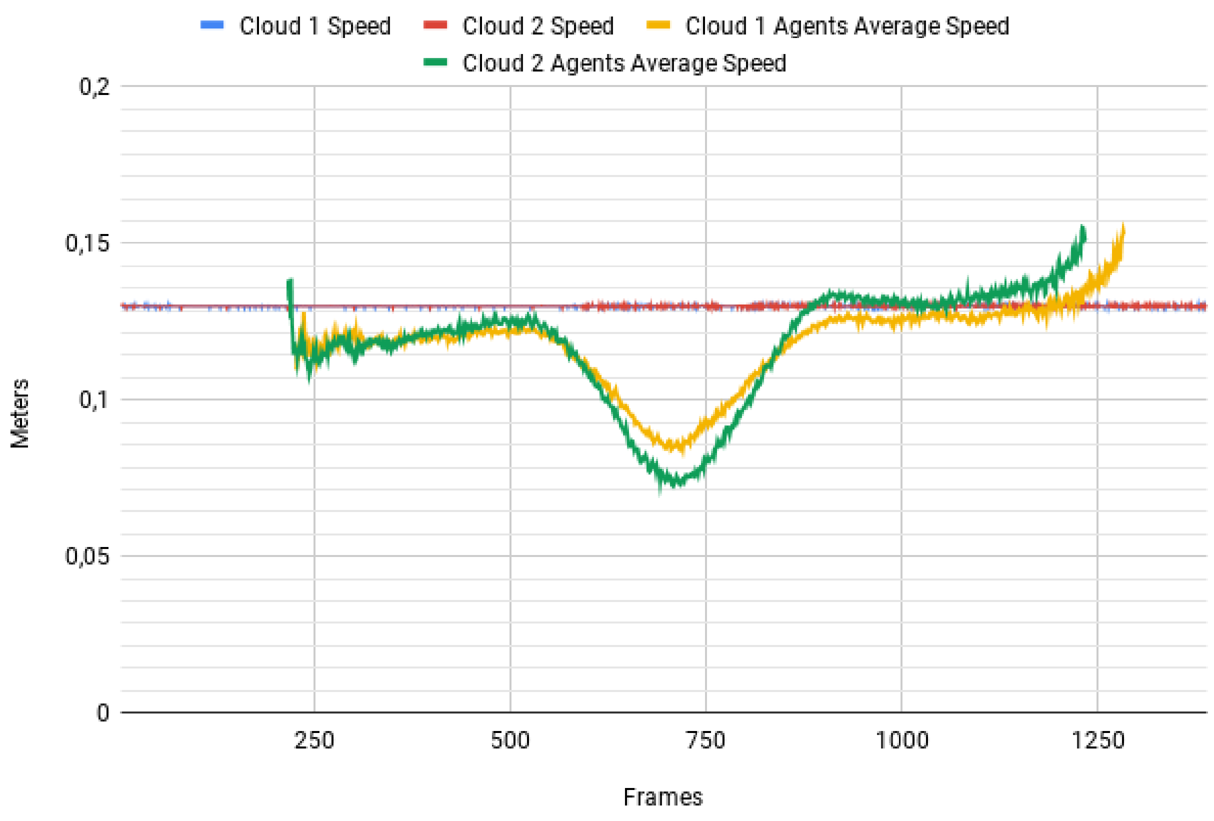


Figure 6.4: The speeds for experiment *B*. Cloud speeds in red and blue. Agent average speeds in green and yellow.

Considering that the perception radius of a cloud increases in situations of space competition, the performance is affected by the densities of each cloud.

To test the performance of BioClouds we designed 4 experiments (*D*, *E*, *F* and *G*), with varying sizes of crowds, with different configurations of number of clouds and cloud sizes. These experiments are simulated utilizing BioClouds only at first, without a MVW, to test the performance of the macroscopic simulation module. The experiments were run with a i7-3770 CPU 3.40GHz, 16GB of RAM and a GeForce GTX 660 GPU.

For the experiments *D*, *E*, *F* and *G*, the environment is a flat square of terrain with no obstacles, measuring 1000 by 1000 meters. Experiments *D* and *E* consider a crowd of 200 thousands agents. Experiments *F* and *G* analyses a crowd of 1 million agents. The preferred density of agents in these experiments is of 3.0 agent$/sqm$, as to represent a densely packed crowd [18]. Experiments *D* and *E* simulate each one of two crowds of 100 thousands agents moving towards each other.

Experiment *D* has *400* clouds of 500 agents each, while experiment *E* has *200* clouds of 1000 agents each. The clouds are spawned in the rectangles delimited by the diagonals $\{(0, 50), (100, 450)\}$ and $\{(400, 50), (500, 450)\}$. Experiments *F* and *G* simulate two crowds of 1 million agents moving towards each other.

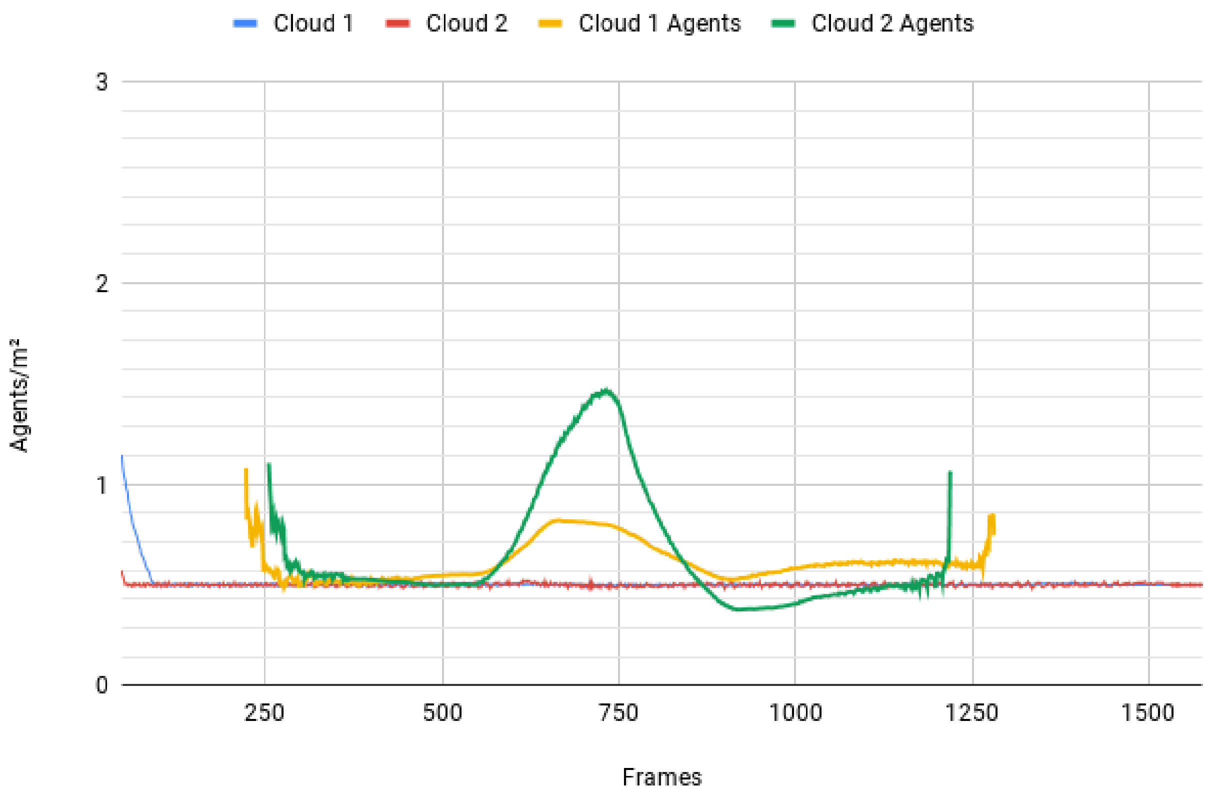


Figure 6.5: The densities for experiment *C*. Cloud densities in red and blue. Agent densities in green and yellow.

Experiment *F* has *2000* clouds of 500 agents each. Experiment *G* has *1000* clouds of 1000 agents each. The clouds are spawned in the rectangles delimited by the diagonals $\{(0,150),(200,850)\}$ and $\{(800,150),(1000,850)\}$.

Since there is variation in time per frame during the simulation, the measure of performance selected is the median of the time per frame during the simulation. Table 6.1 shows the obtained time per frames for BioClouds, and a comparison with the results obtained by Narain et al. [16]. BioClouds is capable of simulating a greater number of agents than the competitive work, with shorter times per frame. In addition, our work shows simulations with a larger number of agents.

Experiments	Agents	Clouds	Cloud Size	Time / Frame
D	100k	400	500	20.80ms
E	100k	200	1000	25.60ms
F	1m	2000	500	86.85ms
G	1m	1000	1000	105.22ms
Narain et al. [16]	100k	-	-	447ms
Narain et al. [16]	80k	-	-	806ms

Table 6.1: Obtained results per experiment and comparison with other studies.

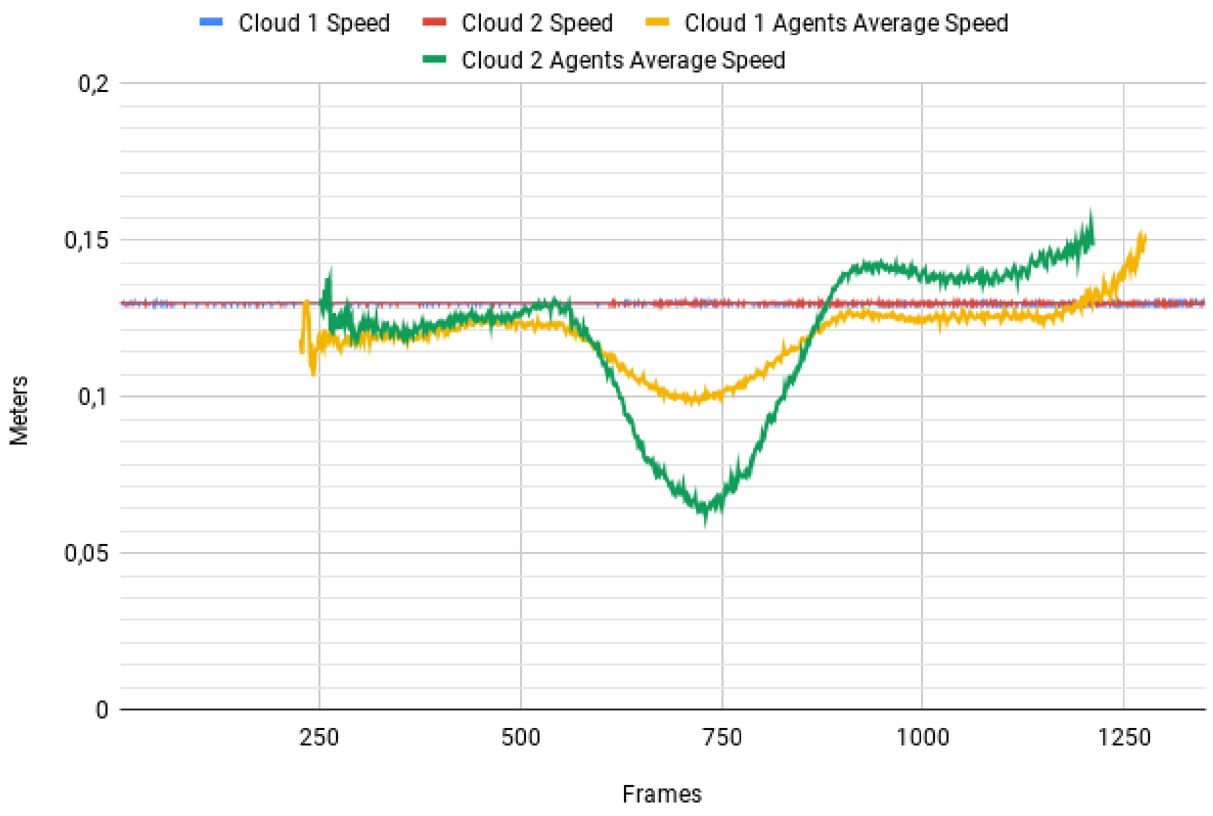


Figure 6.6: The speeds for experiment *C*. Cloud speeds in red and blue. Agent average speeds in green and yellow.

To assess the impact of the MVM during large crowd simulations experiments, *D*, *E*, *F* and *G* were simulated with a with a 25 by 25 meters MVW positioned in the center of the simulated environment. The experiments measured the quantity of agents currently inside the MVW and the time per simulated frame. Table 6.2 shows the median of agents inside the MVW and the median time per frame during the simulated experiments.

Experiments	Agents in MVW	Time / Frame
D	4037	190.94ms
E	2905	166.36ms
F	4866	317.41ms
G	5235	562.96ms

Table 6.2: The median time per frame and median agents in MVW during the execution of experiments *D*, *E*, *F* and *G* with a 25 by 25 meters MVW. The times per frame measured for this analysis ignore frames with no BioCrowds agents.

6.2 Legion Experiments

We now present some experimental results obtained with Legion model. The experiments focus on exploring the new multi-level behavior and the proposed visualization. Experiments *A* and *B* compare how different complexities of environment (varied amount of obstacles) affect split and merge behaviors. In order to compare two different scenarios, the number of split and merge operations were counted during each experiment, as well as the crowd densities and average velocities. Experiments *A* and *B* have the following data:

- The simulation runs at 10 frames per simulated second.
- The cells in the regular grid are $0.125m$ size and area of $0.015625m^2$;
- Each experiment has, at the beginning, one only legion containing 100 agents with preferred density of 1agent/m^2;
- desired speed $S = 1.3m/s$;
- goal position $\vec{G} = (5, 25)$;
- obstacle size $5m \times 5m$, where experiment *A* has one obstacle positioned at $(15, 25)$, and experiment *B* has four obstacles positioned at $(15, 25)$, $(23, 30)$, $(23, 20)$, $(32, 25)$.
- total size of environment $100m \times 50m$.

We want to show the impact of environmental complexity on the emergence of split and merged behaviors. To the best of our knowledge, there is no dynamically hybrid crowd simulation methods which to compare to, so we compare results to the purely macroscopic simulation. In experiment *A*, 1 split and 6 merge operations were executed; while in experiment *B*, 11 split operations and 50 merge operations happened. Table 6.3 compares the two experiments. Merge operations occur more often than split operations since one split operation instantiates 7 legions, which need six merge operations to merge back into one Legion. We emphasize that split/merge operations are asymmetrical, i.e., one Legion can split one time and gives origin to 7 legions, that have to merge in pairs from 4 to 6 times to become the same Legion again.

Table 6.3: results of experiments *A* and *B*: average density (agents/m^2), average speed (m/s), number of splits and merges.

	DENSITY	SPEED	# SPLITS	# MERGES
A	1.001	1.30	1	6
B	1.003	1.29	11	50

Figures 6.9 and 6.11 show snapshots of experiments *A* and *B*: a legion recursively splitting and merging to traverse a scenario with obstacles. In experiment *B*, the first Legion never fully regrouped into a single legion again, since the created small groups got far away from others due

to obstacles. This fact is consistent with real-life: people can drift apart due to space restrictions, and lose group identity. Figure 6.12 shows the same scenario simulated with microscopic BioCrowds behaviors.

It is essential to mention that we decide not to include path planning for all the evaluated simulations, so the only input was the goal and not a path to achieve the goal. We did that because we do not want to interfere with the way the legions will avoid the obstacles, deciding, for instance, going through the right or left side of the obstacles. In order to compare with BioCrowds, we also turned off the path planning, and as a consequence, we can see some agents in a local minimum, behind the obstacles. We compared our results with BioClouds [2], in the environment of experiment *B*, as can be seen in Figure 6.10. We measured the average density (people$/m^2$) along the simulations for the three methods: Legion ($\bar{d} = 1.03$), BioCrowds ($\bar{d} = 1.23$), and BioClouds ($\bar{d} = 0,99$). Even though the results are similar, as expected, BioCrowds achieved higher average density, because at the end agents were very close to each other to achieve the same goal. On the other hand, BioClouds kept lower densities because it is always one circle trying to achieve the desired density. Legion achieved an average density close to the desired one, even after splits and merges.

6.3 LODUS Urban Mobility Experiments

Comparison of the LODUS population dynamics simulation with the state-of-art has one major impediment, to the best of our knowledge there are no comparable simulation models with enough characteristics in common to present a relevant comparison. Additionally, comparison with real world data is very difficult, given how difficult to obtain real world data can be for mobility issues, due to privacy concerns and availability of data by governmental agencies. To this end, we propose using validation metrics, to the degree that a user of the simulation can decide if the generated data is plausible, or for example, a specialist in the urban mobility field could judge and compare with real world data, if he or she finds himself in possession of those data.

To validate the framework and model presented in Chapter 5 and the use case proposed in 5.6.1 we devised six experiments using real world data from the city of Porto Alegre. The experiments are divided into three sets, one set using only rigidly defined population routines, one set using only the Lévy flight model, and a set combining both approaches.

We modeled Porto Alegre in two versions, one reduced to 13 neighborhoods and the full 94 neighborhoods version. The population for each neighborhood was obtained from InLoco[1], a company that collects data from many cities in Brazil. Each neighborhood is modeled as a region containing an assortment of POIs. Each region contains at least the following POIs: *home*, *work*, *school*, *marketplace*, *pharmacy*, and *gas_station*. Regions also may possess some of less common public spaces, for example: *shopping_mall*, *hospital*, and *football stadium*. For this experiment,

[1]https://inloco.com.br/pt/

whether a region has one of the less common POIs, it was decided using real world data for the city of Porto Alegre. The data collection, city description, and generation of the JSON input file was done by Gabriel Fonseca Silva at VHLAB, as part of his Msc. dissertation in the context of the LODUS project. Figure 6.14 shows a representation of the city and the chosen neighborhoods for the performed experiments.

Every simulation was ran with 7 days of 24 steps each, representing hours, for a total of 168 simulation steps. A short simulation duration of a few days is sufficient to analyze short term movement patterns and is small enough for simulation results to be easily parsed visually.

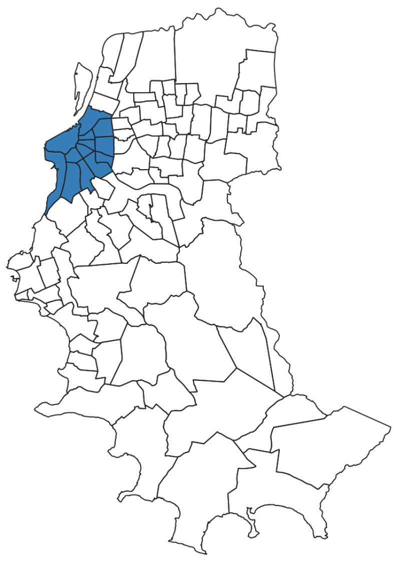


Figure 6.14: The neighborhoods of Porto Alegre. The blue zones are the 13 neighborhoods used in the small environment experiments. The neighborhoods of *Centro*, *Floresta* and *Moinhos de Vento* are contained in the blue colored region.

We focus our analysis on three neighborhoods: *Centro*, *Floresta* and *Moinhos de Vento*. This was done just to reduce informational clutter. For each of the experiments we present a population over time graph, containing the quantity of population in each neighborhood. The graph presents data for local population (which are in the neighborhood in the analyzed time), outsider population (who lives in another neighborhood) and total population (outsider+insider), over the course of the entire simulation. Experiments A and C, with 13 simulated neighborhoods, also present an Origin-Destination Matrix (OD matrix), which characterizes the relationship between each pair of regions. Table 6.4 shows a comparison experiment parameters and goals.

Table 6.4: Table showing the parameters used for each experiment. Numbers in brackets show the set of values compared in that experiment.

Experiment	Neighs.	Routine	Lévy	Restr. (r)	Mov. Prob. (l)	Scale (s)
A	13	x		{0.3, 0.5, 0.7}		
B	94	x		{0.3, 0.5, 0.7}		
C	13		x		0.1	{50, 100}
D	94		x		0.1	{50, 100}
E	13		x		{0.05, 0.1}	50
F	94		x		{0.05, 0.1}	50
G	94	x	x	{0.3, 0.7}	0.1	50
H	94	x	x	0.3	0.1	{50, 100}

6.4 Daily Routines

This section shows our results obtained using a mobility pattern composed of exclusively daily routines [2]. Mobility occurs in two moments, during frames 7 and 18 of the daily loop. At hour 7 all workers will move from their *home* POIs to *work* POIs. This distribution is weighted according to the distance between both regions. At hour 18, every worker goes back to their original *home* POIs. This routine also has a movement restriction parameter r, this parameter reduces the quantity of workers which leave *home* at hour 7 linearly. New quantity $q' = q \times (1 - r)$, where q is the original quantity of workers which would be moved. We used values for r as follows: 0.3, 0.5 and 0.7 for the movement restriction parameters, which approximate the average movement pattern for Porto Alegre before the pandemic, the average movement pattern during the pandemic and the maximum verified restriction of movement[3], respectively.

Experiments A and B describe simulations for 13 and 94 neighborhoods and are shown respectively in Figure 6.15 and Figure 6.16. It is possible to see how the rigidly defined routine presents no daily variation.

Figures 6.18, 6.19 and 6.20 show the OD matrices for experiment A, with 13 neighborhoods, for all 3 values for r. The OD matrix shows only the population which moved outside their native regions, as such, shows no restriction of movement. The ratios of the population distribution is not affected by the restriction parameters. Figure 6.17 shows snapshots of experiment A, using a Legion [1] inspired visualization.

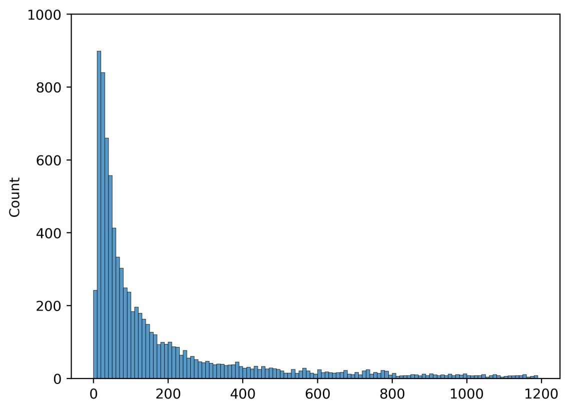
6.5 Stochastic Behavior

To solve the issue of rigidity and determinism of the simulation, we implemented the Lévy Flight [15] model in LODUS. This behavior randomly moves population between POIs according to

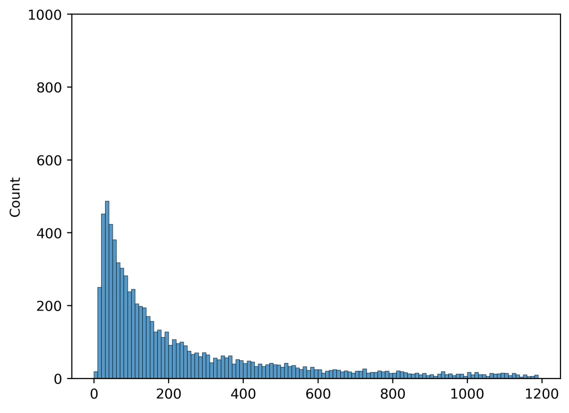
[2]The daily routine description was created by Gabriel Fonseca Silva using data from InLoco.

[3]According to the InLoco data.

a Lévy distribution function. This operation was defined as a Global Routine with cycle length $= 1$. At each simulation step, each POI separates their population in population packets, for each packet p the POI samples a travel distance according to a Lévy distribution[4] with location 0 and scale s, and moves the population in p to a POI in that distance bin (as described in Section 5.6.1). The parameter l determines what is the probability of a POI to sample a distance for any packet p. For these experiments we used values 50 and 100 for s, 10 for the size of p and values 0.05 and 0.1 for l. Figure 6.21 shows a histogram of values for two Lévy distributions, one with scale 50 and one with scale 100. The stochastic movement introduced by adding a Lévy flight behavior to the population does not form a routine for a city, but it serves as a way to make the simulation more dynamic and lifelike.



(a) Distribution with scale $= 50$.



(b) Distribution with scale $= 100$.

Figure 6.21: A Levy distribution. Compares the effects of two scales on the shape of a Lévy distribution. A larger scale flattens the distribution and increases weight of the tail.

Experiments C and D are shown in Figure 6.22 and Figure 6.23, and show the simulations for 13 and 94 neighborhoods, respectively, with both values of sampling scale s. The probability l to sample a packet was 0.1. It is possible to see how the random walk slowly moves population to other regions and the effect of changing the scale of the Lévy sampling on the characteristic of the generated population.

Figures 6.24 and 6.25 show the OD matrices for experiment C, with 13 neighborhoods, for both values of s. The OD matrices show different short term migration patterns that form with a different probability distribution defining Lévy jumps. Changing scale value changes the preference of individuals to travel to farther or closer places, this is reflected on the emergent migration pattern.

Experiments E and F are shown in Figure 6.26 and Figure 6.27, and show the simulations for 13 and 94 neighborhoods, respectively, with both values of probability l. The value for s used was 50. It is possible to see how the speed of population dispersion by the random walk actions changes with the probability for a packet to perform a Lévy jump.

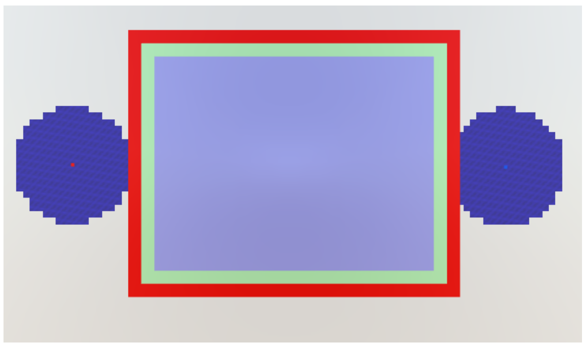
[4]We used the Scipy python module Lévy distribution implementation. It offers two control parameters, location and scale, to control the shape and location of the distribution.

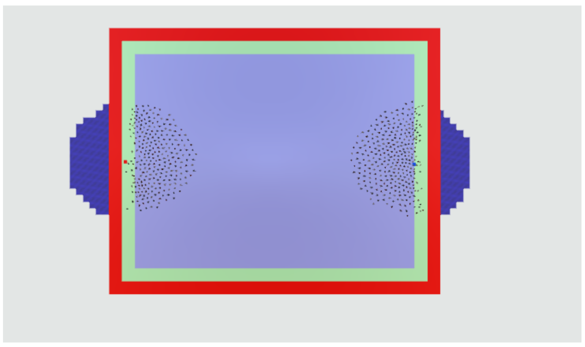
6.6 Daily Routines and Stochastic Behavior

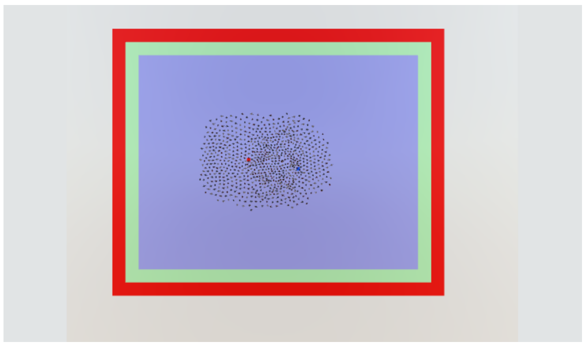
Combining both simulation approaches, daily routines and stochastic walk, we can model a population which has a defined set of favorite spots but also visits, albeit less often, the different facilities around the city. Experiment G and H combine the routine movement behavior of Experiments A and B, with the Lévy walk mobility of experiments C, D, E and F. Experiment G compares the effect of two values of movement restriction, $r = 0.3$ and $r = 0.7$, a Lévy movement behavior with scale $s = 50$. Experiment H compares the effect of two values of Lévy movement scale, $s = 50$ and $s = 100$.

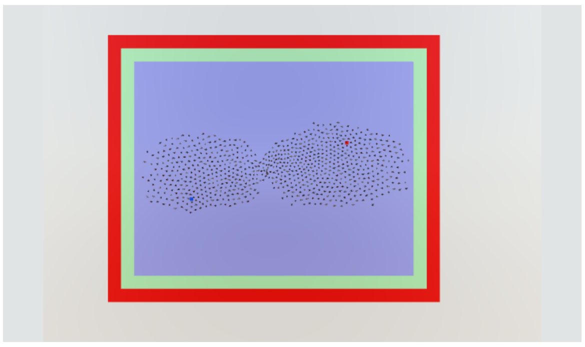
Figure 6.28 the population overtime graph of the combined movement patterns, over the three selected neighborhoods, for experiment G. It shows the dominant recurring and deterministic routine, with the added noisy daily variation of the Lévy flight movement pattern. Experiment G compares values for $r = 0.3$ and $r = 0.7$.

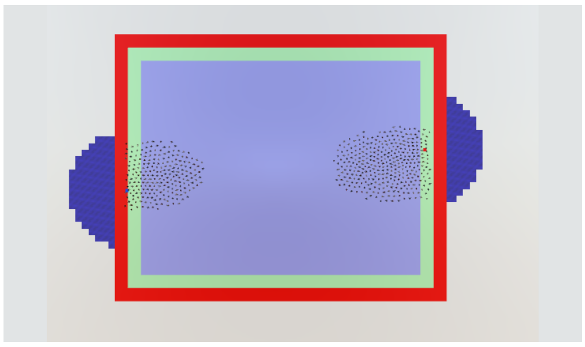
Figure 6.29 the population overtime graph of the combined movement patterns, over the three selected neighborhoods, for experiment H. It shows the dominant recurring and deterministic routine, with the added noisy daily variation of the Lévy flight movement pattern. Experiment G compares values for $s = 50$ and $s = 100$.


(a) Two clouds walking towards opposite directions.


(b) Agents being spawned when clouds enter the MVW.


(c) Agents crossing each other.


(d) Agents crossed each other.


(e) Agents leaving the MVW and clouds resurfacing.

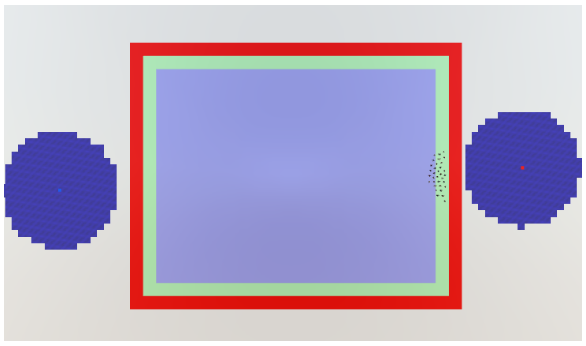

(f) Agents left the MVW, just clouds remaining.

Figure 6.7: Snapshots from Experiment B. Two clouds (blue circles) want to reach their respective goals, in opposite directions (a). Then, when they enter inside the MVW area (red rectangle), agents begin to be spawned (b). In (c), it is possible to see agents from both clouds forming lanes while crossing each other. When agents finish to cross each other (d), they keep following their respective paths until they leave the MVW area, which makes the clouds resurface (e). Finally, agents left the MVW area, so only the clouds continue to be simulated (f).

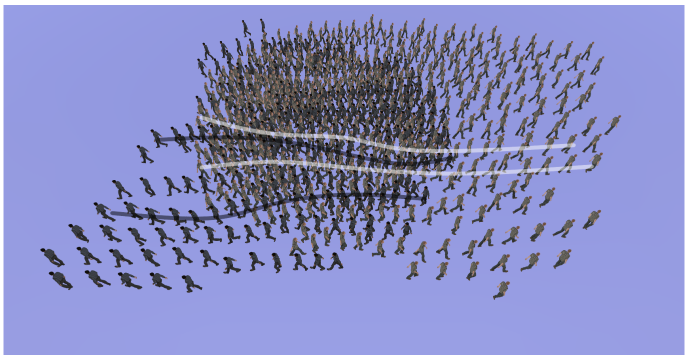


Figure 6.8: The zoomed image of Figure 6.7c. Showing the formation of lanes in the MVW. The lines overlaying agents represents the paths representing the emergent lanes. Grey lanes are walking left. Light-grey lanes are walking right.

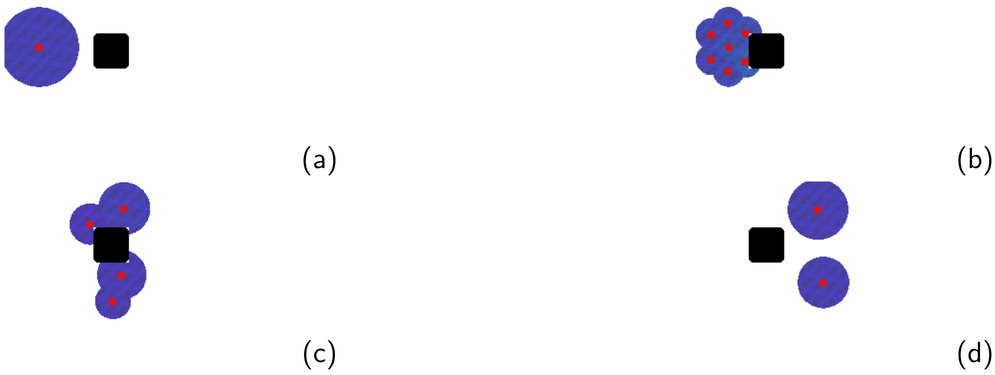


Figure 6.9: Snapshots from experiment *A* using the heatmap visualization. (a) A legion heads towards an obstacle. (b) The legion splits due to the obstacle. (c) The legions go around the obstacle. (d) The legions begin regrouping.

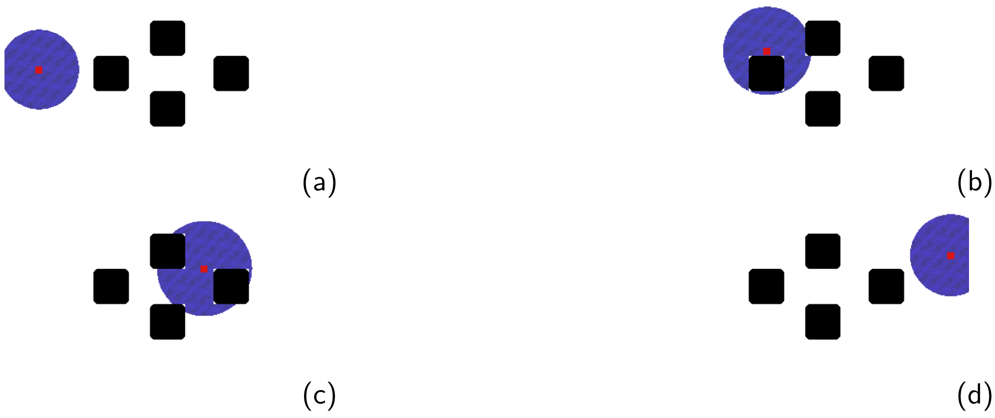


Figure 6.10: Snapshots from experiment *B*, simulated with BioClouds. (a) A cloud heads towards an obstacle. (b) The cloud avoids the obstacle and increases its perception radius. (c) The cloud avoids another obstacle. (d) The cloud moves towards the goal and returns to its original size.

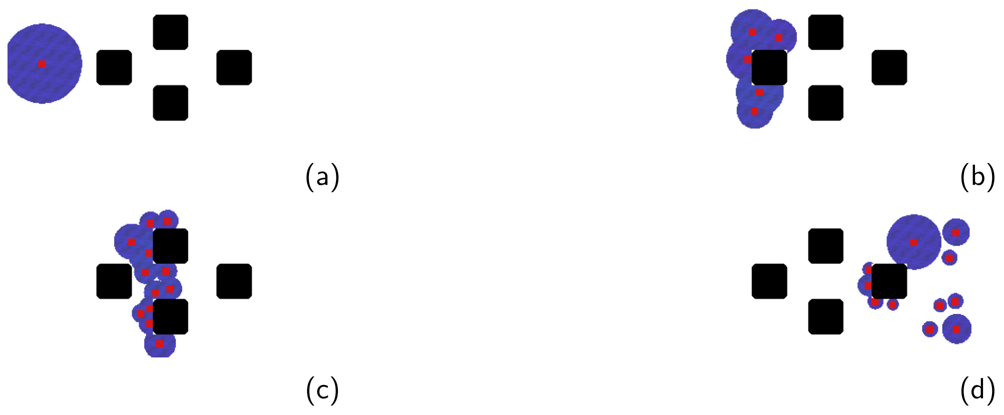


Figure 6.11: Snapshots from experiment *B* using the heatmap visualization. (a) A legion heads towards the obstacles. (b) The legion splits due to the obstacles. (c) The legions go around the obstacles. (d) The legions begin regrouping.

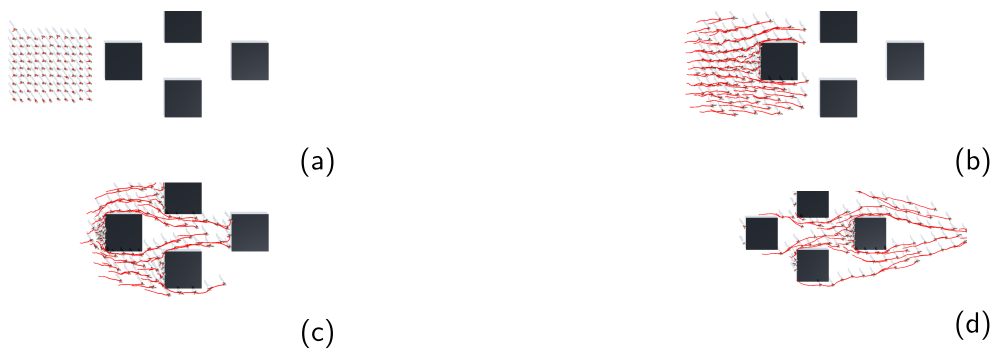


Figure 6.12: Snapshots of experiment *B* simulated with BioCrowds. A crowd of BioCrowds agents separating and regrouping when interacting with the obstacles. (a) A crowd splits when interacting with an obstacle. (b) The crowd goes around the obstacle. (c) The crowds start grouping back together. (d) The crowds continue grouping back together.

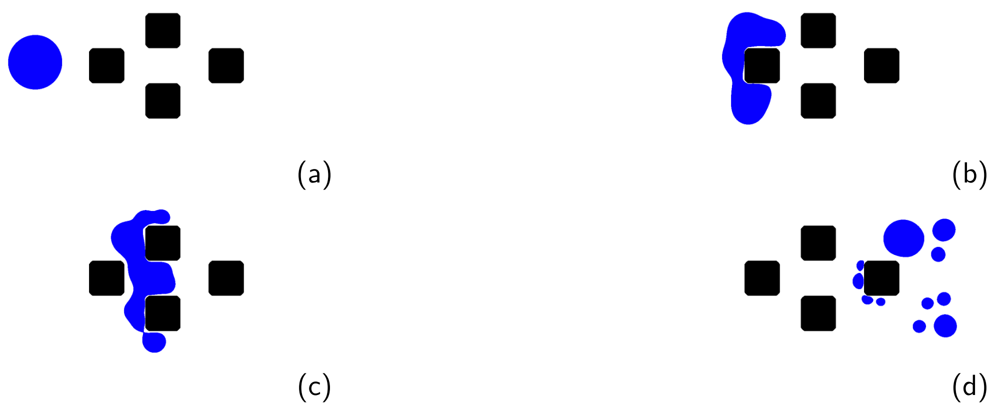
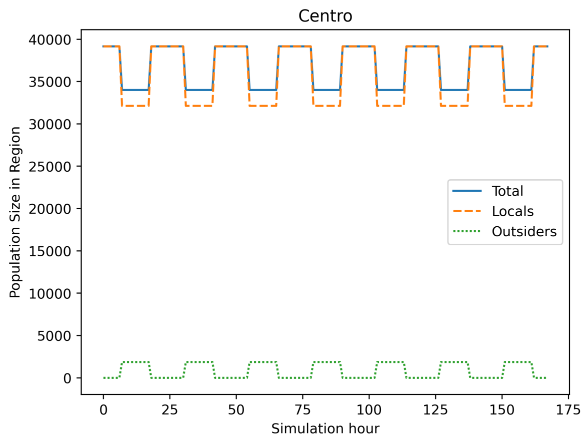
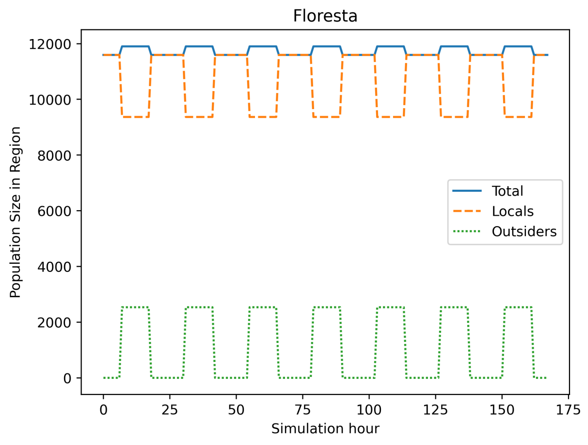


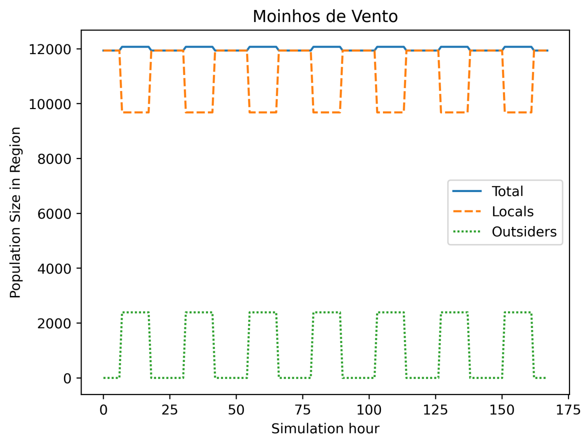
Figure 6.13: Snapshots from experiment *B* using our blobby visualization. (a) A legion heads towards the obstacles. (b) The legion splits due to the obstacles. (c) The legions go around the obstacles. (d) The legions begin regrouping.



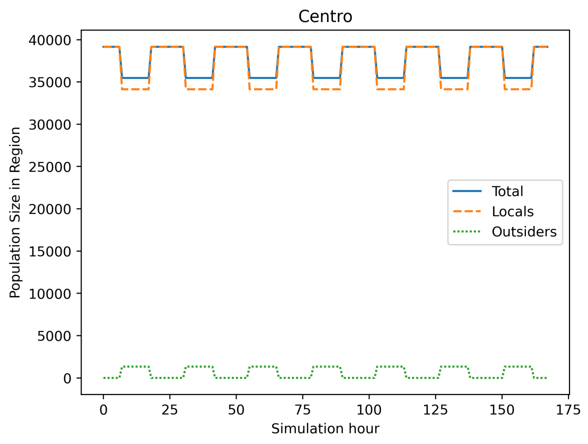
(a) Population over time with $r = 0.3$ in neighborhood Centro.



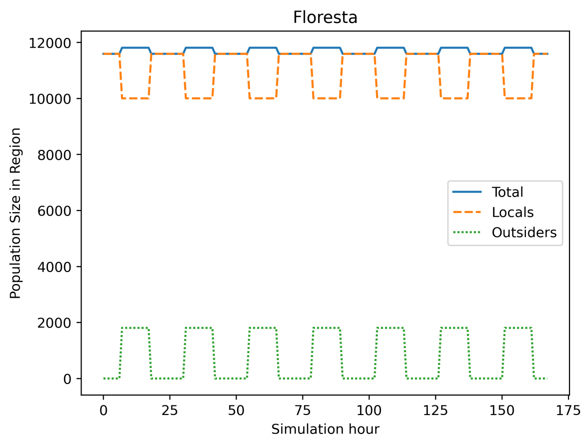
(b) Population over time with $r = 0.3$ in neighborhood Floresta.



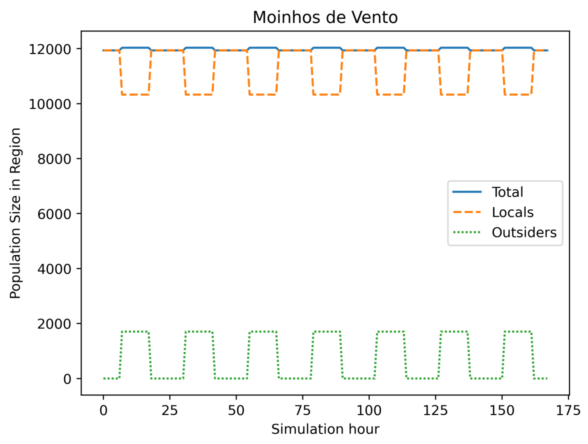
(c) Population over time with $r = 0.3$ in neighborhood Moinhos de Vento.



(d) Population over time with $r = 0.5$ in neighborhood Centro.

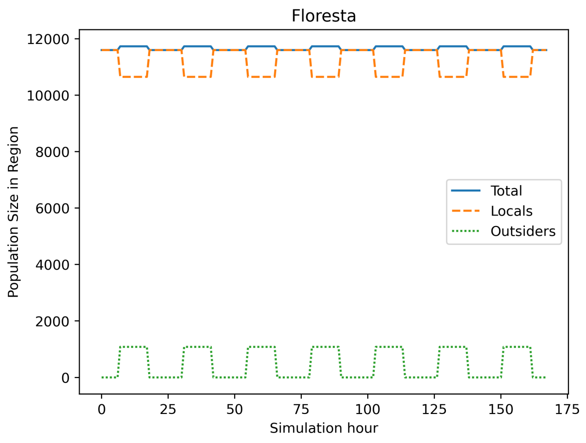


(e) Population over time with $r = 0.5$ for neighborhood Floresta.

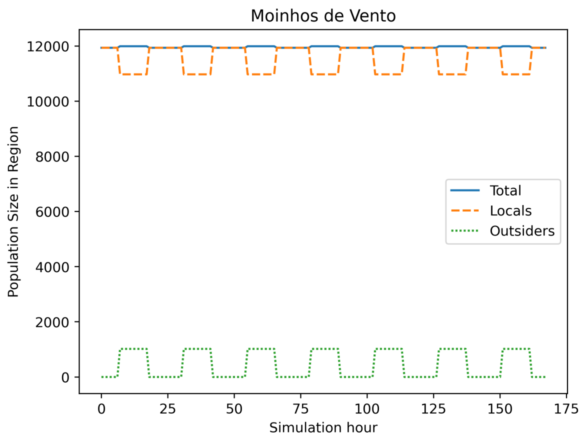


(f) Population over time with $r = 0.3$ for neighborhood Moinhos de Vento.

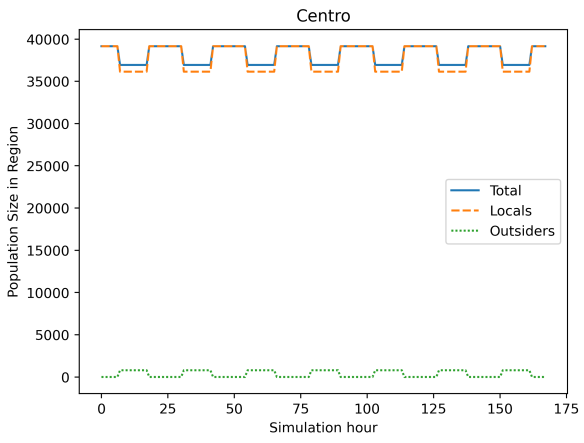
Figure 6.15: Experiment A: Population over time at each of the analyzed neighborhoods. Using routine movement description and comparing the effect of different movement restriction (r) values. Simulation performed in a model city with 13 neighborhoods.



(g) Population over time with $r = 0.7$ for neighborhood Floresta.

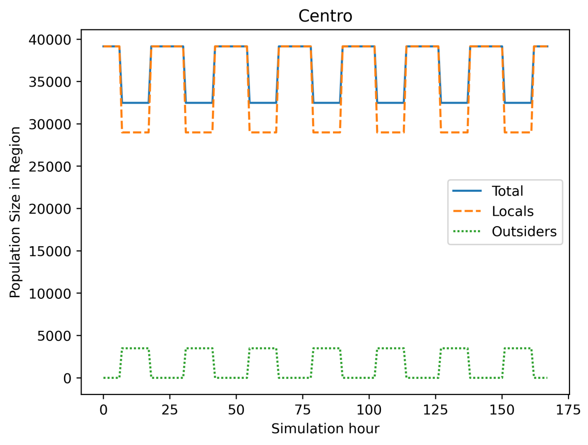


(h) Population over time $r = 0.7$ for neighborhood Moinhos de Vento.

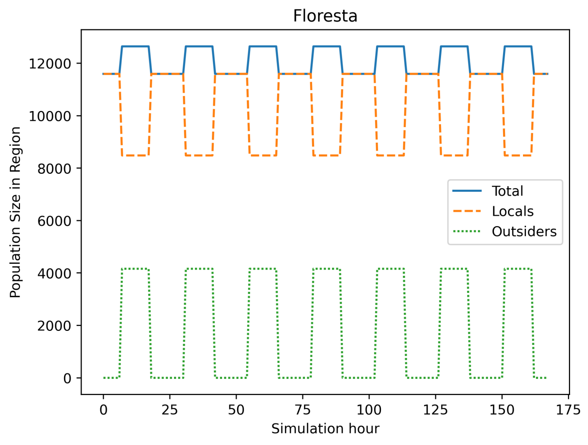


(i) Population over time with $r = 0.7$ for neighborhood Centro.

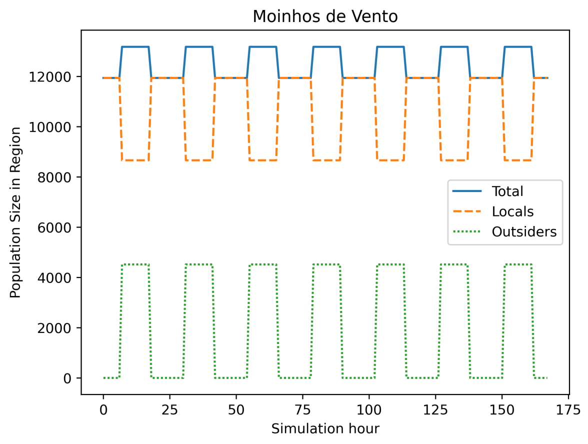
Figure 6.15: Cont.



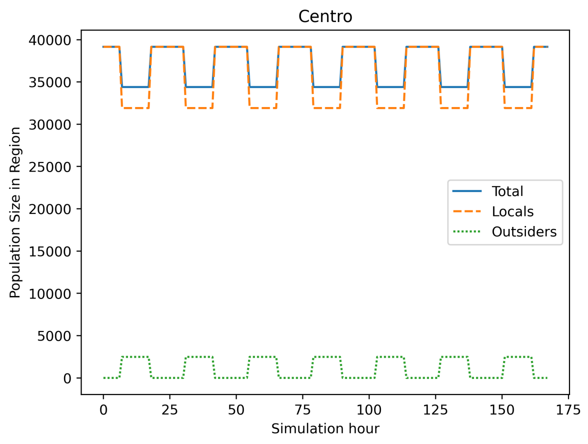
(a) Population over time with $r = 0.3$ in neighborhood Centro.



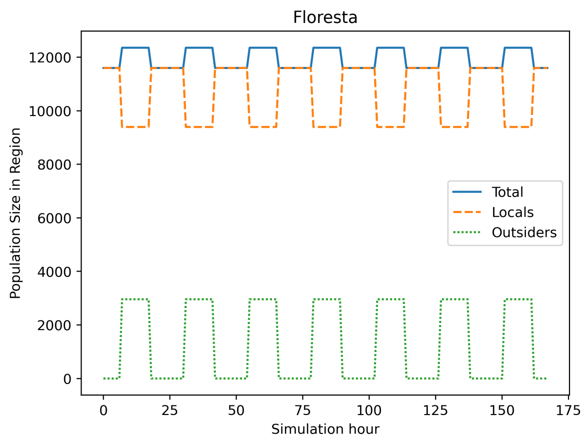
(b) Population over time with $r = 0.3$ in neighborhood Floresta.



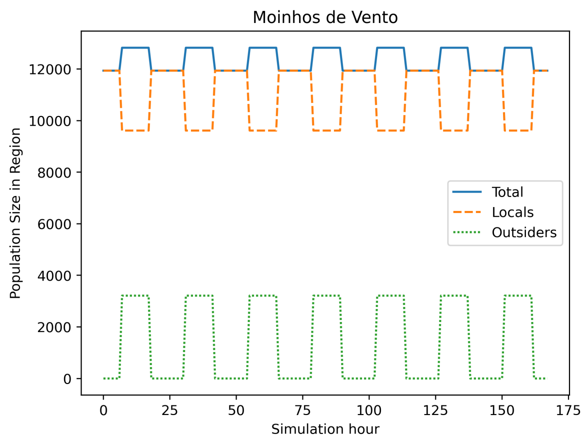
(c) Population over time with $r = 0.3$ in neighborhood Moinhos de Vento.



(d) Population over time with $r = 0.5$ in neighborhood Centro.

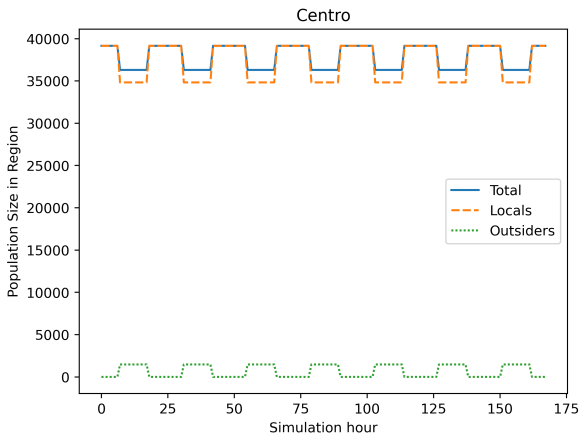


(e) Population over time with $r = 0.5$ in neighborhood Floresta.

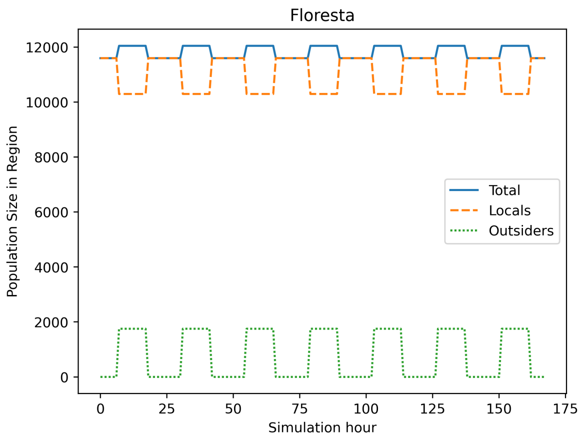


(f) Population over time with $r = 0.5$ in neighborhood Moinhos de Vento.

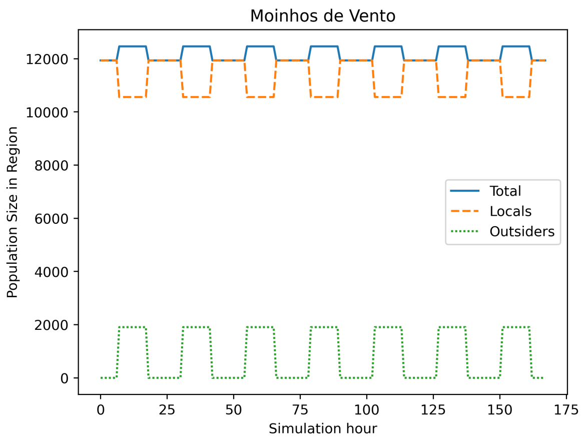
Figure 6.16: Experiment B: Population over time in each of the analyzed neighborhoods. Using routine movement description and comparing the effects of using movement restriction (r) values $0.3, 0.5, 0.7$. On a model city with 94 neighborhoods.



(g) Population over time with $r = 0.7$ in neighborhood Centro.

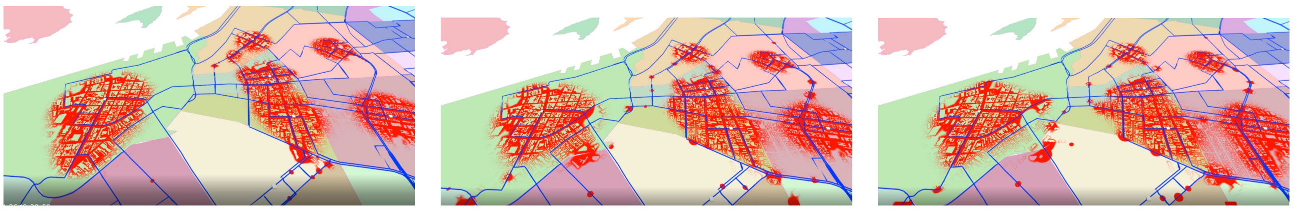


(h) Population over time with $r = 0.7$ in neighborhood Foresta.



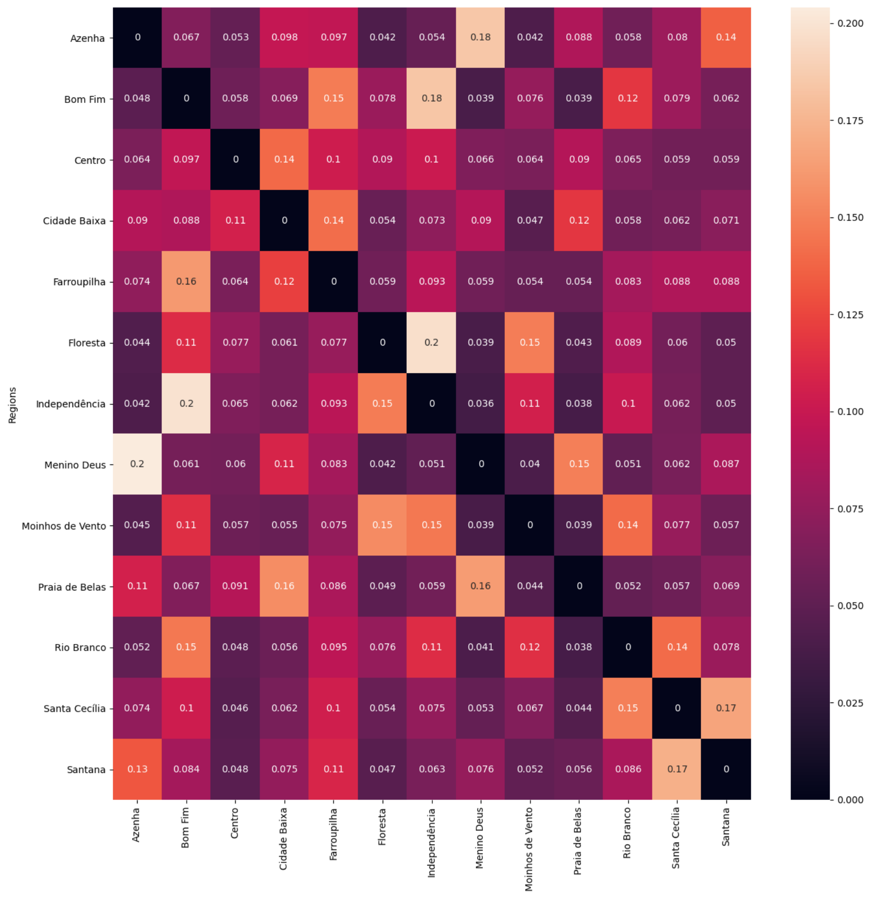
(i) Population over time with $r = 0.7$ in neighborhood Moinhos de Vento.

Figure 6.16: Cont.



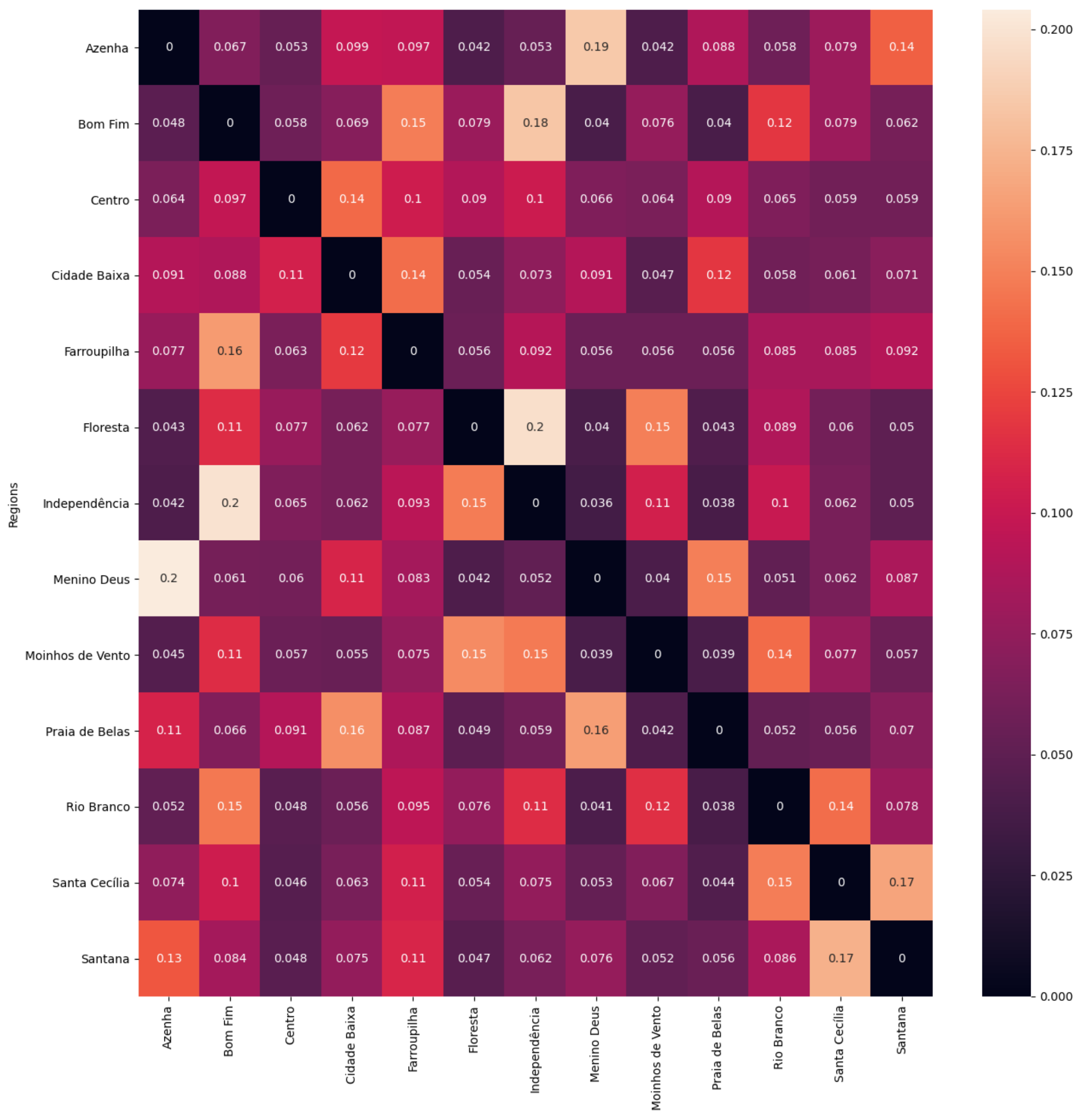
(a) (b) (c)

Figure 6.17: Three moments of Experiment A visualized with a Legion based visualization approach. This visualization uses a distance functions to model buildings.



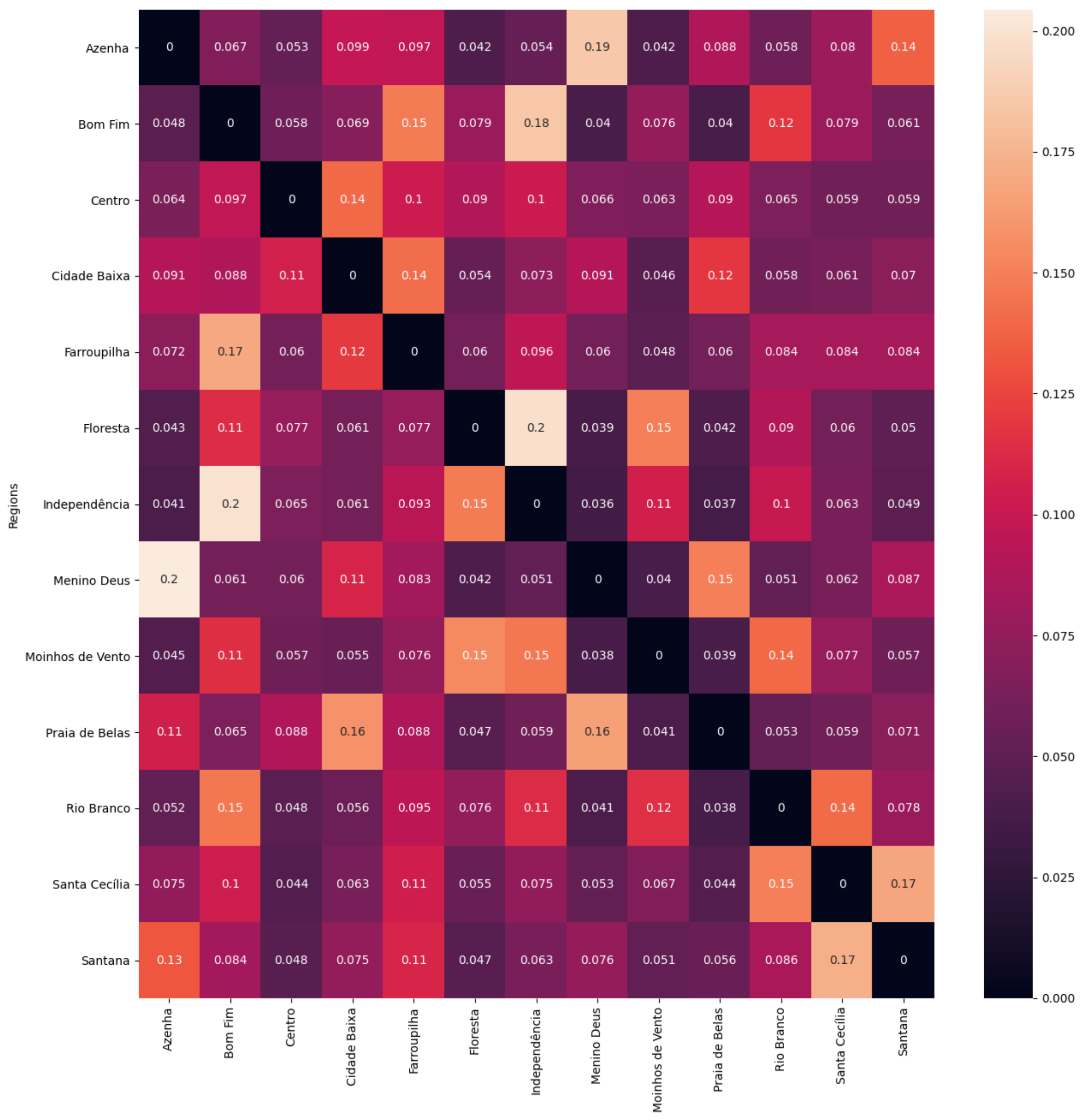
	Azenha	Bom Fim	Centro	Cidade Baixa	Farroupilha	Floresta	Independência	Menino Deus	Moinhos de Vento	Praia de Belas	Rio Branco	Santa Cecília	Santana
Azenha	0	0.067	0.053	0.098	0.097	0.042	0.054	0.18	0.042	0.088	0.058	0.08	0.14
Bom Fim	0.048	0	0.058	0.069	0.15	0.078	0.18	0.039	0.076	0.039	0.12	0.079	0.062
Centro	0.064	0.097	0	0.14	0.1	0.09	0.1	0.066	0.064	0.09	0.065	0.059	0.059
Cidade Baixa	0.09	0.088	0.11	0	0.14	0.054	0.073	0.09	0.047	0.12	0.058	0.062	0.071
Farroupilha	0.074	0.16	0.064	0.12	0	0.059	0.093	0.059	0.054	0.054	0.083	0.088	0.088
Floresta	0.044	0.11	0.077	0.061	0.077	0	0.2	0.039	0.15	0.043	0.089	0.06	0.05
Independência	0.042	0.2	0.065	0.062	0.093	0.15	0	0.036	0.11	0.038	0.1	0.062	0.05
Menino Deus	0.2	0.061	0.06	0.11	0.083	0.042	0.051	0	0.04	0.15	0.051	0.062	0.087
Moinhos de Vento	0.045	0.11	0.057	0.055	0.075	0.15	0.15	0.039	0	0.039	0.14	0.077	0.057
Praia de Belas	0.11	0.067	0.091	0.16	0.086	0.049	0.059	0.16	0.044	0	0.052	0.057	0.069
Rio Branco	0.052	0.15	0.048	0.056	0.095	0.076	0.11	0.041	0.12	0.038	0	0.14	0.078
Santa Cecília	0.074	0.1	0.046	0.062	0.1	0.054	0.075	0.053	0.067	0.044	0.15	0	0.17
Santana	0.13	0.084	0.048	0.075	0.11	0.047	0.063	0.076	0.052	0.056	0.086	0.17	0

Figure 6.18: Origin-Destination matrix for Experiment A with $r = 0.3$.The diagonal contains only zero values as we chose to represent only the population which traveled outside their native regions.



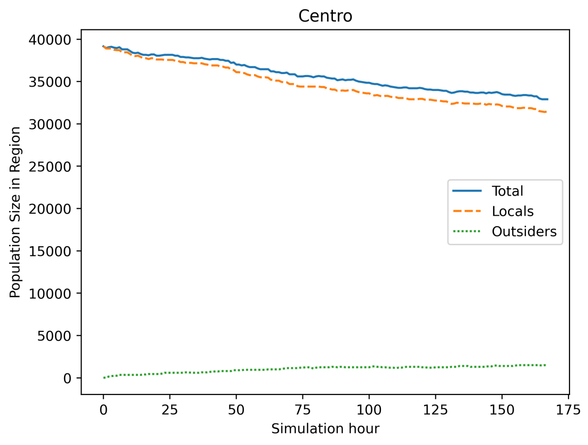
Regions	Azenha	Bom Fim	Centro	Cidade Baixa	Farroupilha	Floresta	Independência	Menino Deus	Moinhos de Vento	Praia de Belas	Rio Branco	Santa Cecília	Santana
Azenha	0	0.067	0.053	0.099	0.097	0.042	0.053	0.19	0.042	0.088	0.058	0.079	0.14
Bom Fim	0.048	0	0.058	0.069	0.15	0.079	0.18	0.04	0.076	0.04	0.12	0.079	0.062
Centro	0.064	0.097	0	0.14	0.1	0.09	0.1	0.066	0.064	0.09	0.065	0.059	0.059
Cidade Baixa	0.091	0.088	0.11	0	0.14	0.054	0.073	0.091	0.047	0.12	0.058	0.061	0.071
Farroupilha	0.077	0.16	0.063	0.12	0	0.056	0.092	0.056	0.056	0.056	0.085	0.085	0.092
Floresta	0.043	0.11	0.077	0.062	0.077	0	0.2	0.04	0.15	0.043	0.089	0.06	0.05
Independência	0.042	0.2	0.065	0.062	0.093	0.15	0	0.036	0.11	0.038	0.1	0.062	0.05
Menino Deus	0.2	0.061	0.06	0.11	0.083	0.042	0.052	0	0.04	0.15	0.051	0.062	0.087
Moinhos de Vento	0.045	0.11	0.057	0.055	0.075	0.15	0.15	0.039	0	0.039	0.14	0.077	0.057
Praia de Belas	0.11	0.066	0.091	0.16	0.087	0.049	0.059	0.16	0.042	0	0.052	0.056	0.07
Rio Branco	0.052	0.15	0.048	0.056	0.095	0.076	0.11	0.041	0.12	0.038	0	0.14	0.078
Santa Cecília	0.074	0.1	0.046	0.063	0.11	0.054	0.075	0.053	0.067	0.044	0.15	0	0.17
Santana	0.13	0.084	0.048	0.075	0.11	0.047	0.062	0.076	0.052	0.056	0.086	0.17	0

Figure 6.19: Origin-Destination matrix for Experiment A with $r = 0.5$.The diagonal contains only zero values as we chose to represent only the population which traveled outside their native regions.

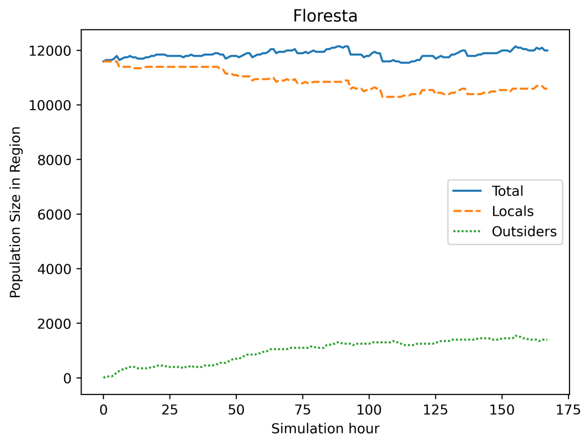


	Azenha	Bom Fim	Centro	Cidade Baixa	Farroupilha	Floresta	Independência	Menino Deus	Moinhos de Vento	Praia de Belas	Rio Branco	Santa Cecília	Santana
Azenha	0	0.067	0.053	0.099	0.097	0.042	0.054	0.19	0.042	0.088	0.058	0.08	0.14
Bom Fim	0.048	0	0.058	0.069	0.15	0.079	0.18	0.04	0.076	0.04	0.12	0.079	0.061
Centro	0.064	0.097	0	0.14	0.1	0.09	0.1	0.066	0.063	0.09	0.065	0.059	0.059
Cidade Baixa	0.091	0.088	0.11	0	0.14	0.054	0.073	0.091	0.046	0.12	0.058	0.061	0.07
Farroupilha	0.072	0.17	0.06	0.12	0	0.06	0.096	0.06	0.048	0.06	0.084	0.084	0.084
Floresta	0.043	0.11	0.077	0.061	0.077	0	0.2	0.039	0.15	0.042	0.09	0.06	0.05
Independência	0.041	0.2	0.065	0.061	0.093	0.15	0	0.036	0.11	0.037	0.1	0.063	0.049
Menino Deus	0.2	0.061	0.06	0.11	0.083	0.042	0.051	0	0.04	0.15	0.051	0.062	0.087
Moinhos de Vento	0.045	0.11	0.057	0.055	0.076	0.15	0.15	0.038	0	0.039	0.14	0.077	0.057
Praia de Belas	0.11	0.065	0.088	0.16	0.088	0.047	0.059	0.16	0.041	0	0.053	0.059	0.071
Rio Branco	0.052	0.15	0.048	0.056	0.095	0.076	0.11	0.041	0.12	0.038	0	0.14	0.078
Santa Cecília	0.075	0.1	0.044	0.063	0.11	0.055	0.075	0.053	0.067	0.044	0.15	0	0.17
Santana	0.13	0.084	0.048	0.075	0.11	0.047	0.063	0.076	0.051	0.056	0.086	0.17	0

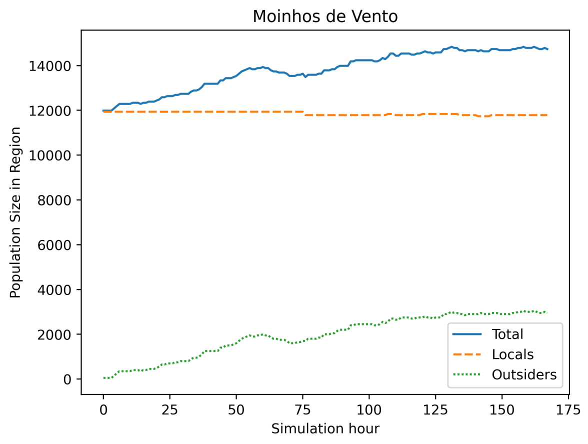
Figure 6.20: Origin-Destination matrix for Experiment A with $r = 0.7$.The diagonal contains only zero values as we chose to represent only the population which traveled outside their native regions.



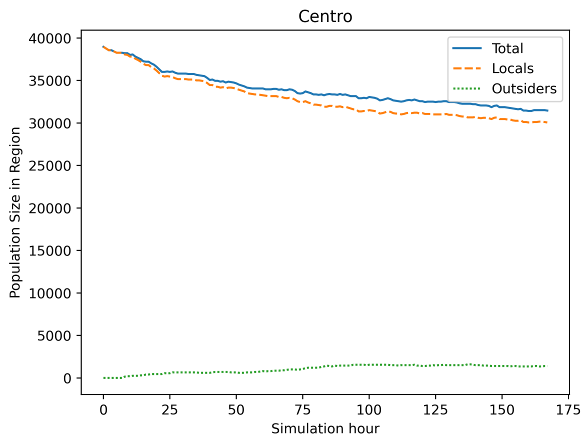
(a) Population over time with
$s = 50$ in neighborhood Centro.



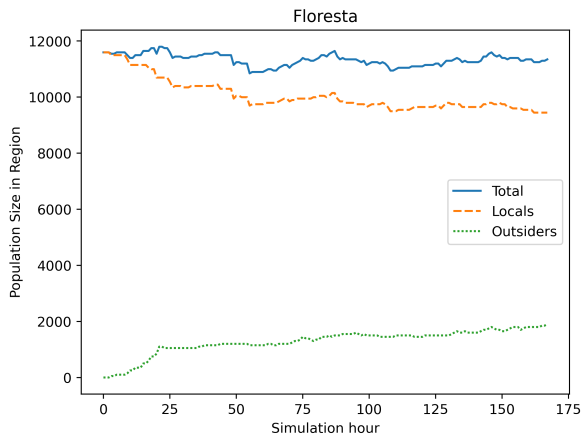
(b) Population over time with
$s = 50$ in neighborhood Floresta.



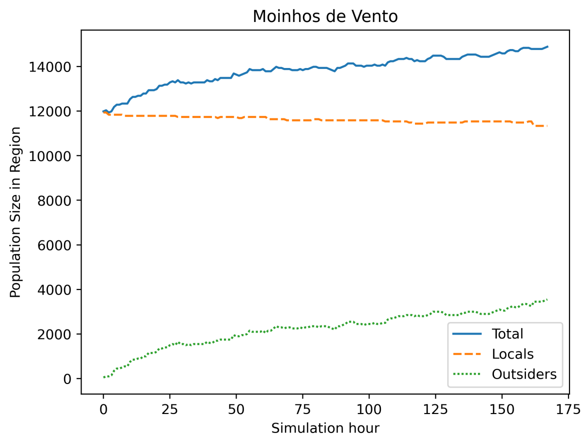
(c) Population over time with
$s = 50$ in neighborhood Moinhos de Vento.



(d) Population over time with
$s = 100$ in neighborhood Centro.

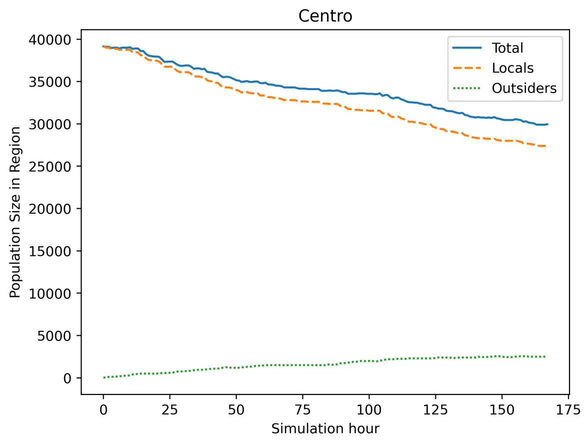


(e) Population over time with
$s = 100$ in neighborhood Floresta.

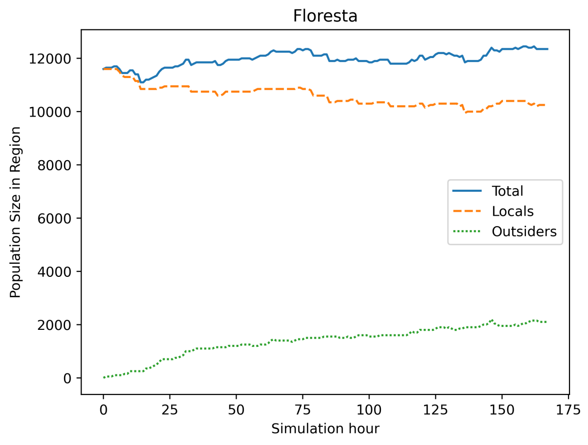


(f) Population over time with
$s = 100$ in neighborhood Moinhos de Vento.

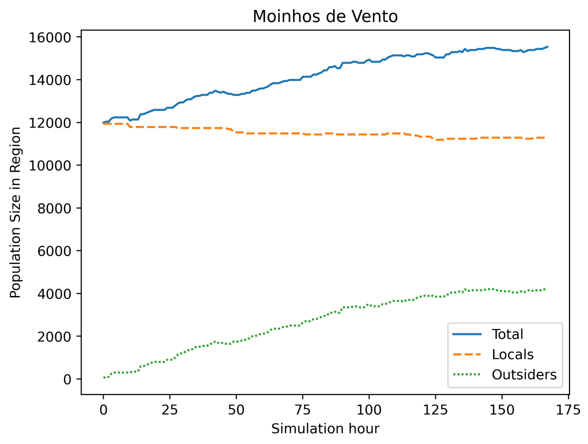
Figure 6.22: Experiment C: Population over time at each of the analyzed neighborhoods. Using the Lévy distribution movement description and comparing the effects of using scale (s) values $50, 100$. On a model city with 13 neighborhoods.



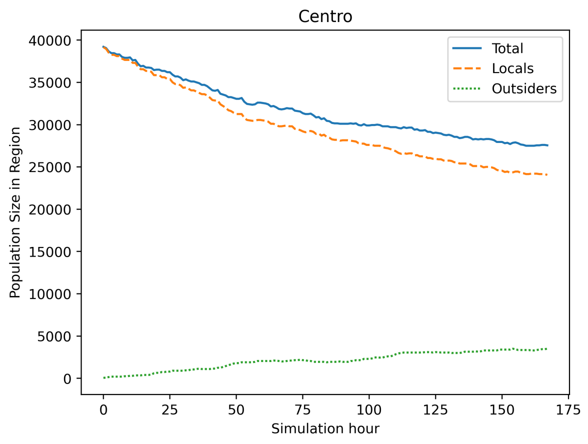
(a) Population over time with $s = 50$ in neighborhood Centro.



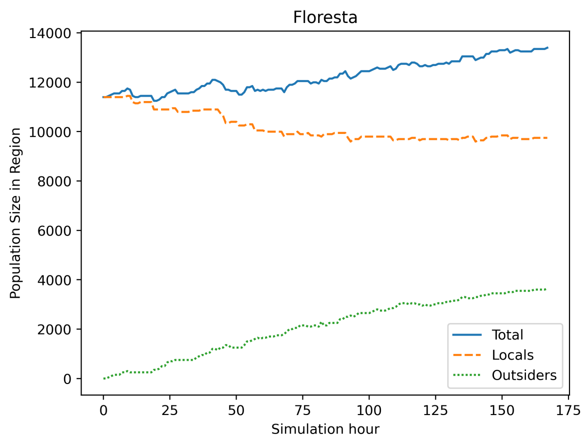
(b) Population over time with $s = 50$ in neighborhood Floresta.



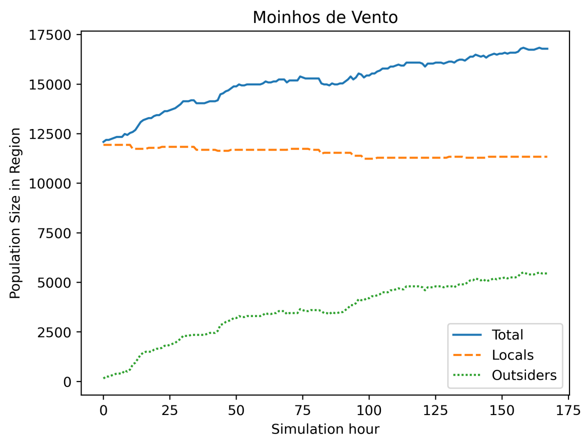
(c) Population over time with $s = 50$ in neighborhood Moinhos de Vento.



(d) Population over time with $s = 100$ in neighborhood Centro.

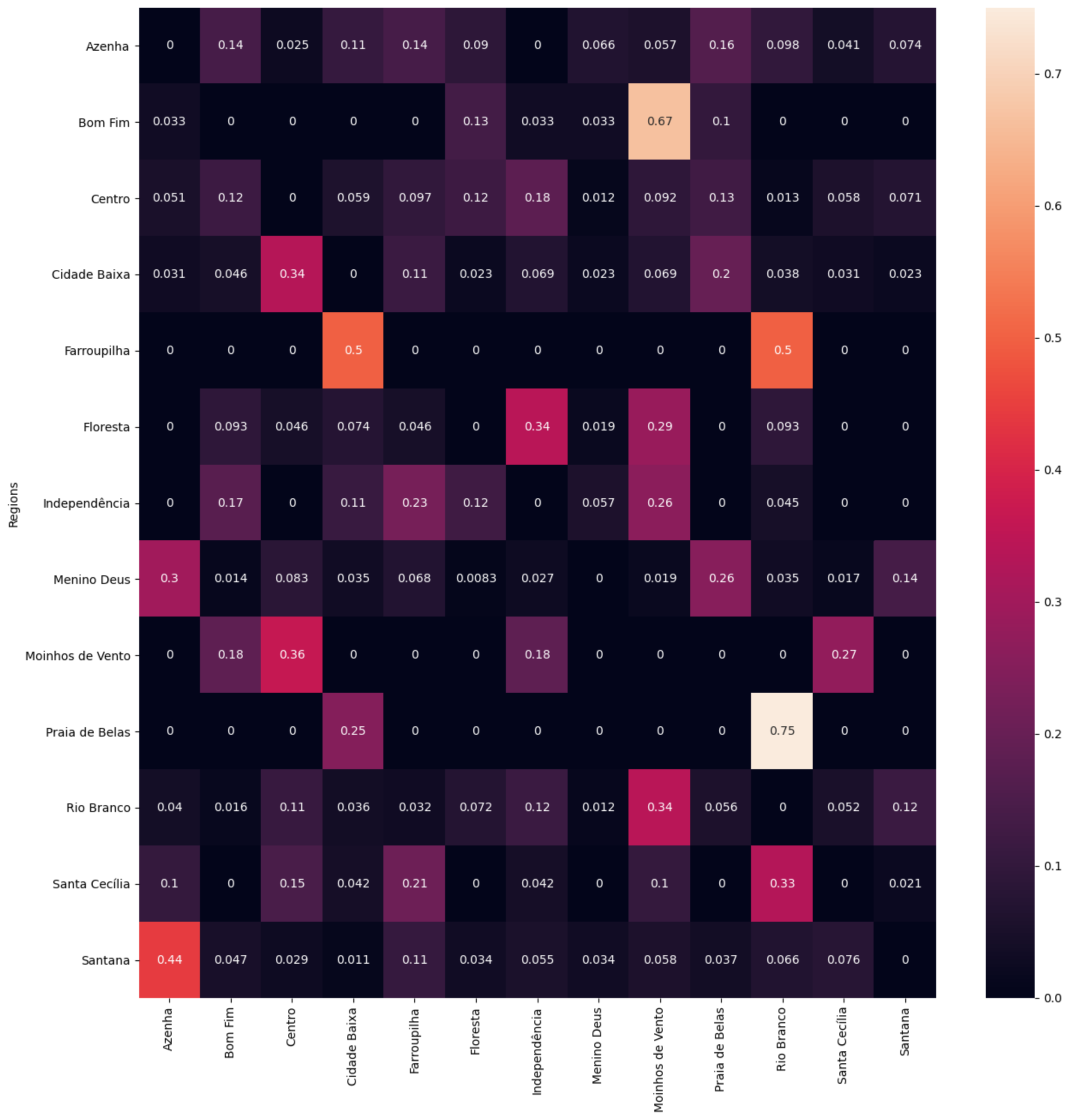


(e) Population over time with $s = 100$ in neighborhood Floresta.



(f) Population over time with $s = 100$ in neighborhood Moinhos de Vento.

Figure 6.23: Experiment D: Population over time in each of the analyzed neighborhoods. Using the Lévy distribution movement description and comparing the effects of using scale values $50, 100$. Simulation performed in a city with 94 neighborhoods.



	Azenha	Bom Fim	Centro	Cidade Baixa	Farroupilha	Floresta	Independência	Menino Deus	Moinhos de Vento	Praia de Belas	Rio Branco	Santa Cecília	Santana
Azenha	0	0.14	0.025	0.11	0.14	0.09	0	0.066	0.057	0.16	0.098	0.041	0.074
Bom Fim	0.033	0	0	0	0	0.13	0.033	0.033	0.67	0.1	0	0	0
Centro	0.051	0.12	0	0.059	0.097	0.12	0.18	0.012	0.092	0.13	0.013	0.058	0.071
Cidade Baixa	0.031	0.046	0.34	0	0.11	0.023	0.069	0.023	0.069	0.2	0.038	0.031	0.023
Farroupilha	0	0	0	0.5	0	0	0	0	0	0	0.5	0	0
Floresta	0	0.093	0.046	0.074	0.046	0	0.34	0.019	0.29	0	0.093	0	0
Independência	0	0.17	0	0.11	0.23	0.12	0	0.057	0.26	0	0.045	0	0
Menino Deus	0.3	0.014	0.083	0.035	0.068	0.0083	0.027	0	0.019	0.26	0.035	0.017	0.14
Moinhos de Vento	0	0.18	0.36	0	0	0	0.18	0	0	0	0	0.27	0
Praia de Belas	0	0	0	0.25	0	0	0	0	0	0	0.75	0	0
Rio Branco	0.04	0.016	0.11	0.036	0.032	0.072	0.12	0.012	0.34	0.056	0	0.052	0.12
Santa Cecília	0.1	0	0.15	0.042	0.21	0	0.042	0	0.1	0	0.33	0	0.021
Santana	0.44	0.047	0.029	0.011	0.11	0.034	0.055	0.034	0.058	0.037	0.066	0.076	0

Figure 6.24: Origin-Destination matrix for Experiment C with $s = 50$.The diagonal contains only zero values as we chose to represent only the population which traveled outside their native regions.

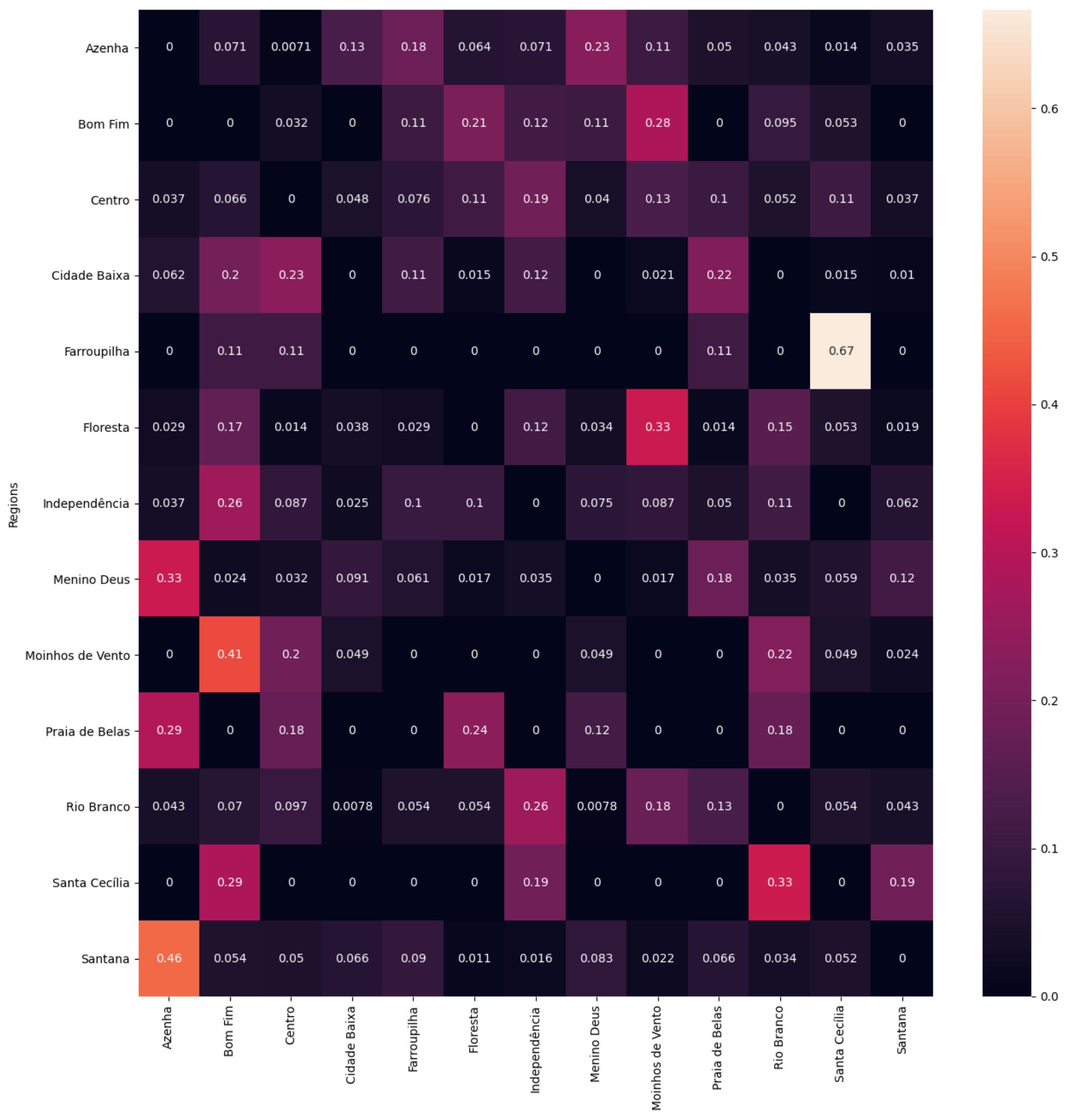
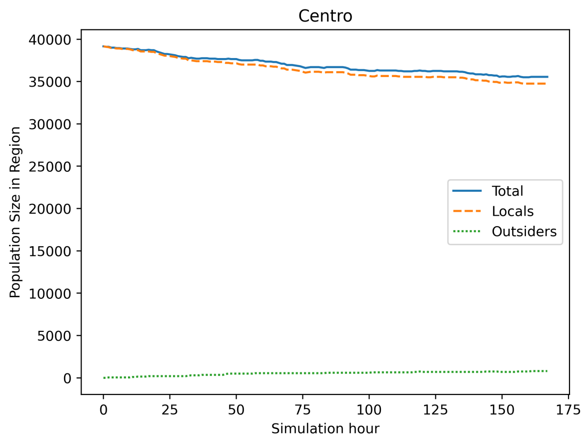
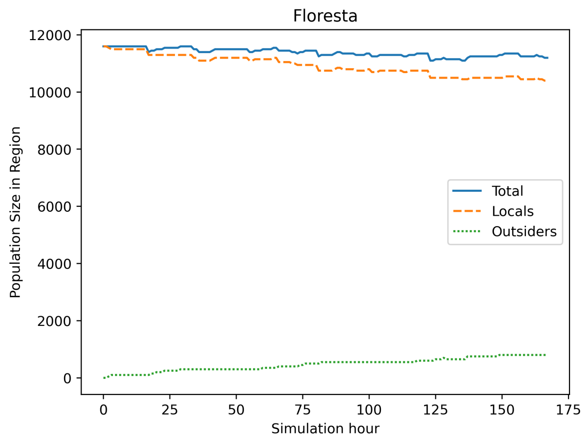


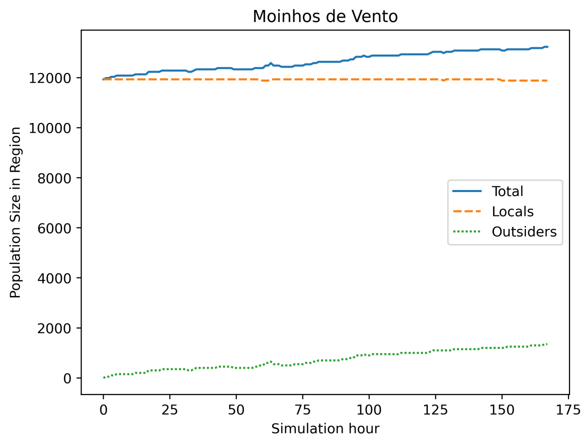
Figure 6.25: Origin-Destination matrix for Experiment C with $s = 100$. The diagonal contains only zero values as we chose to represent only the population which traveled outside their native regions.



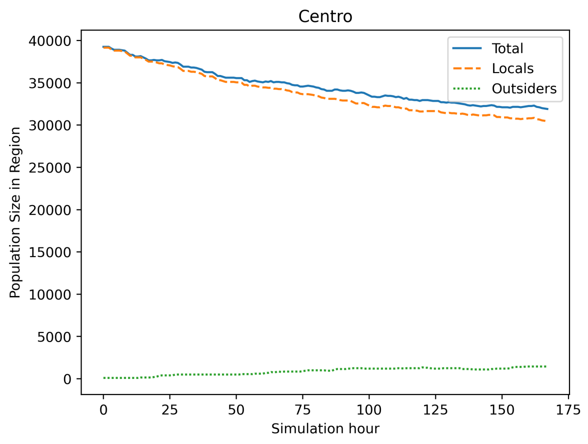
(a) Population over time with $l = 0.05$ in neighborhood Centro.



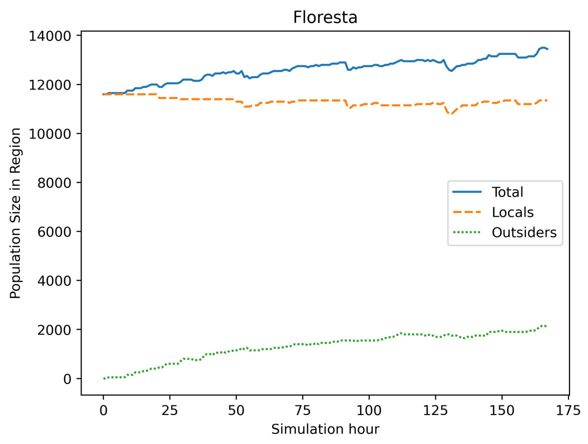
(b) Population over time with $l = 0.05$ in neighborhood Floresta.



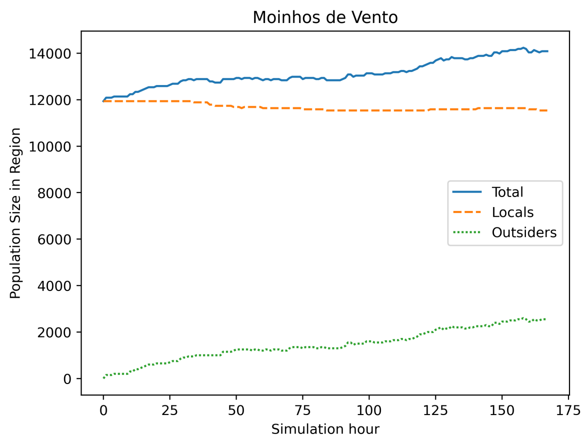
(c) Population over time with $l = 0.05$ in neighborhood Moinhos de Vento.



(d) Population over time with $l = 0.1$ in neighborhood Centro.

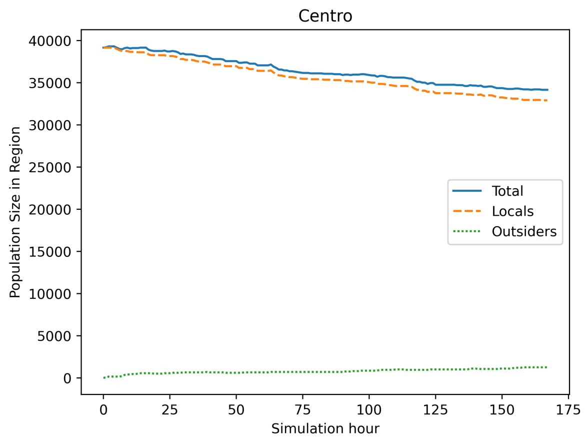


(e) Population over time with $l = 0.1$ in neighborhood Floresta.

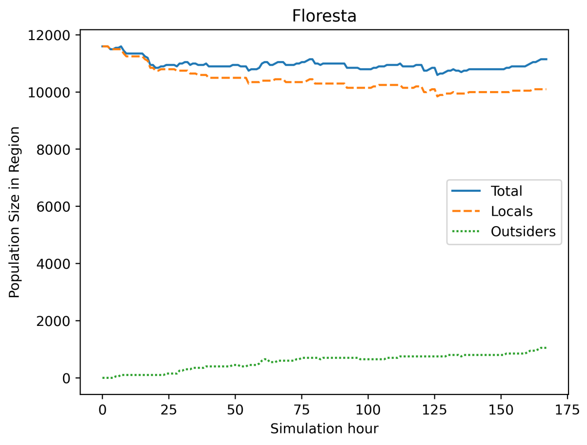


(f) Population over time with $l = 0.1$ in neighborhood Moinhos de Vento.

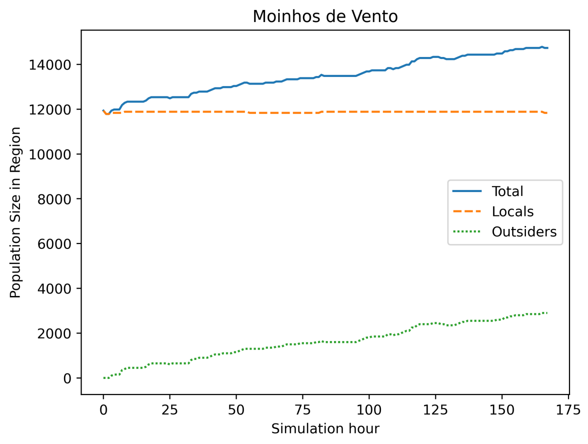
Figure 6.26: Experiment E: Population over time in each of the analyzed neighborhoods. Using the Lévy distribution movement description and comparing the effects of using different mobility probability (l) values. On a model city with 13 neighborhoods.



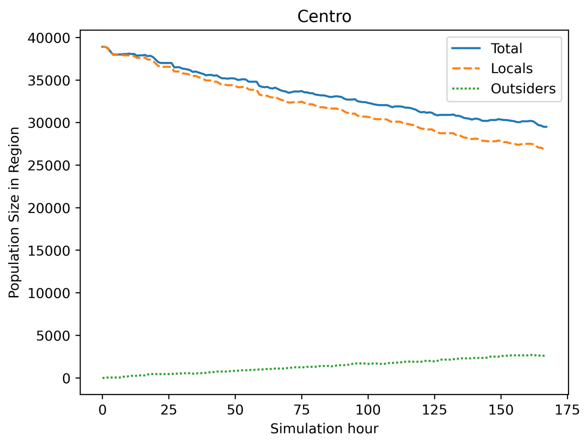
(a) Population over time with $l = 0.05$ in neighborhood Centro.



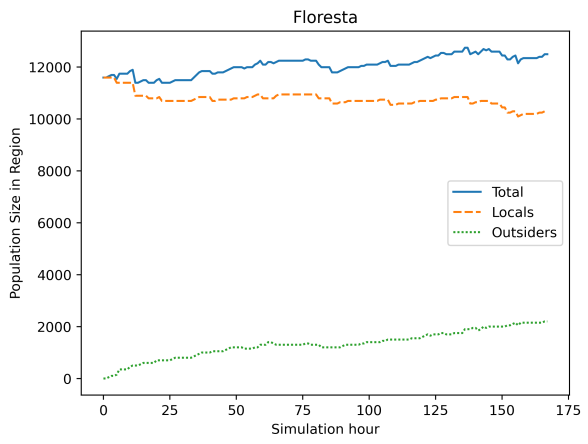
(b) Population over time with $l = 0.05$ in neighborhood Floresta.



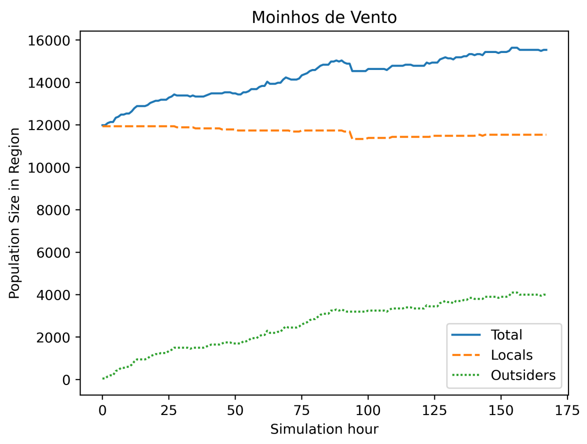
(c) Population over time with $l = 0.05$ in neighborhood Moinhos de Vento.



(d) Population over time with $l = 0.1$ in neighborhood Centro.

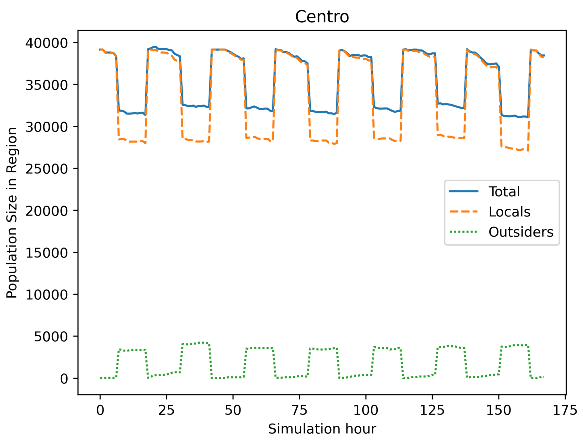


(e) Population over time with $l = 0.1$ in neighborhood Floresta.

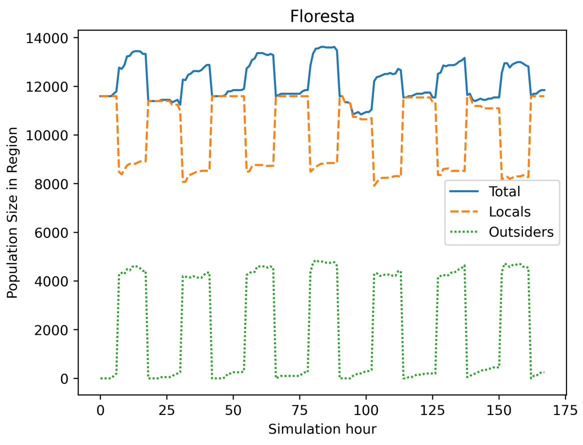


(f) Population over time with $l = 0.1$ in neighborhood Moinhos de Vento.

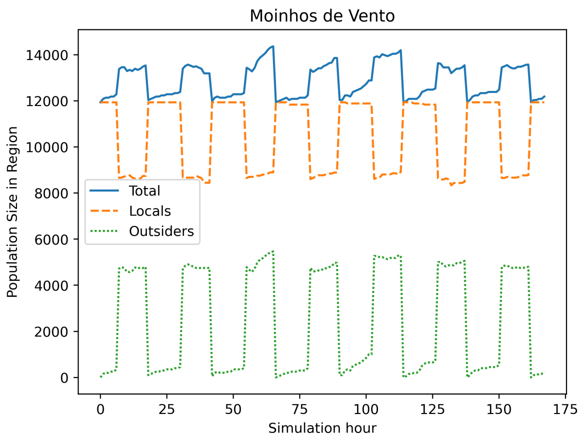
Figure 6.27: Experiment F: Population over time in each of the analyzed neighborhoods. Using the Lévy distribution movement description and comparing the effects of using different mobility probability (l) values. On a model city with 13 neighborhoods.



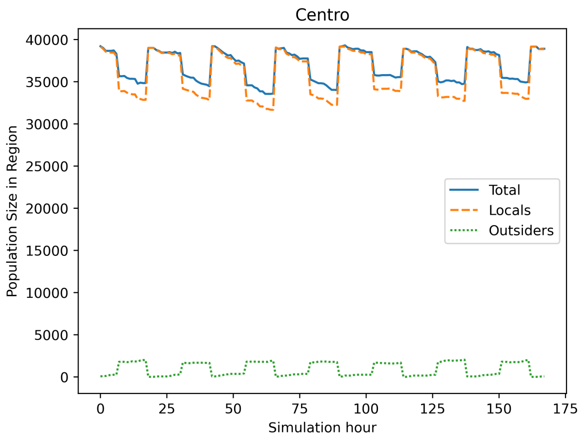
(a) Population over time in Centro with $r = 0.3$.



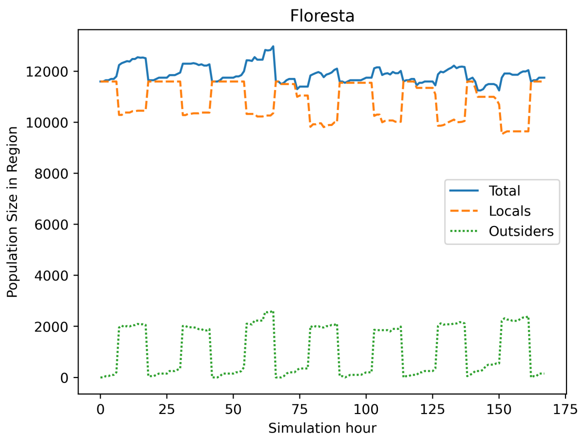
(b) Population over time in Floresta with $r = 0.3$.



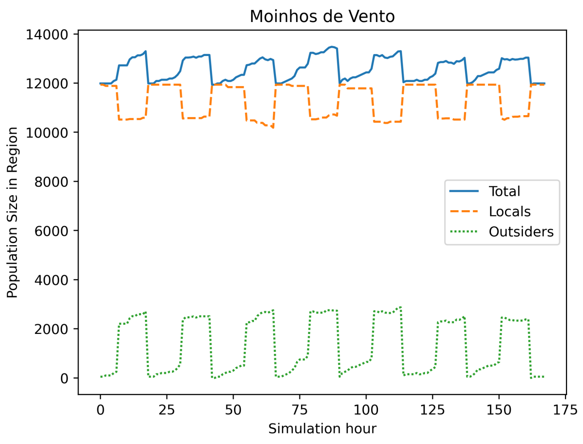
(c) Population over time in Moinhos de Vento with $r = 0.3$.



(d) Population over time in Centro with $r = 0.7$.

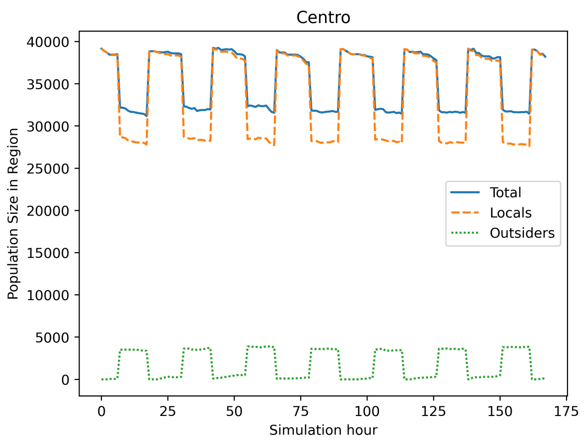


(e) Population over time in Floresta with $r = 0.7$.

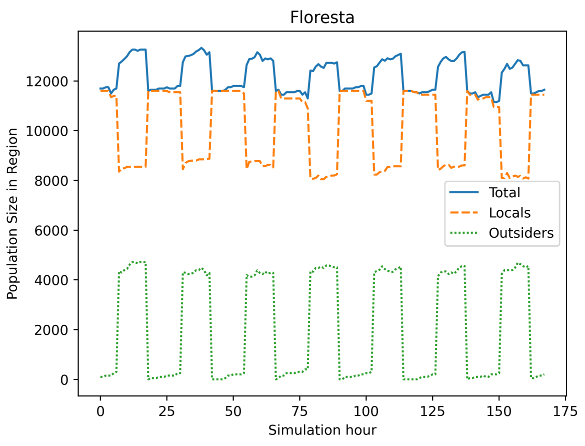


(f) Population over time in Moinhos de Vento with $r = 0.7$.

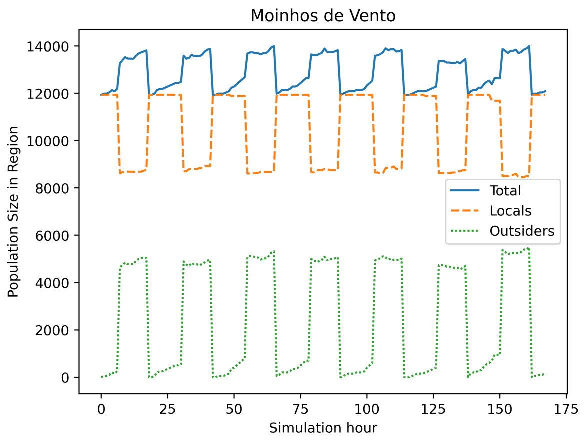
Figure 6.28: Experiment G: Population over time in each of the analyzed neighborhoods. Combining the routine and Lévy distribution approaches to movement description. With Lévy movement behavior scale $s = 50$. The graphs compare movement restrictions of $r = 0.3$ and $r = 0.7$.



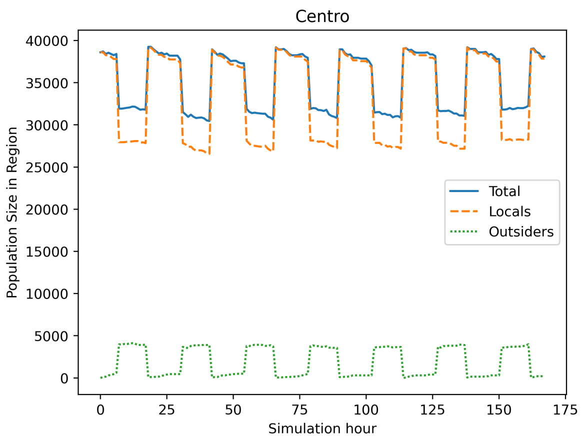
(a) Population over time in Centro with $s = 50$.



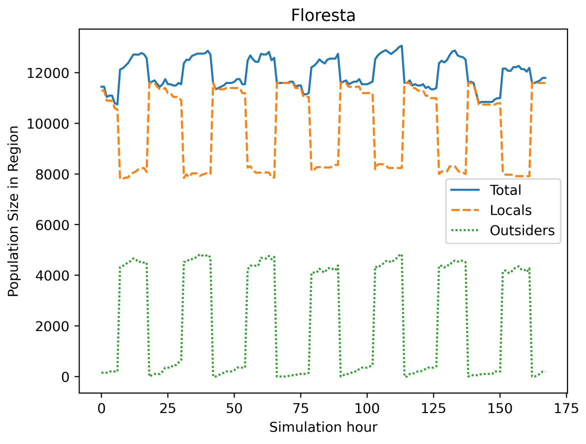
(b) Population over time in Floresta with $s = 50$.



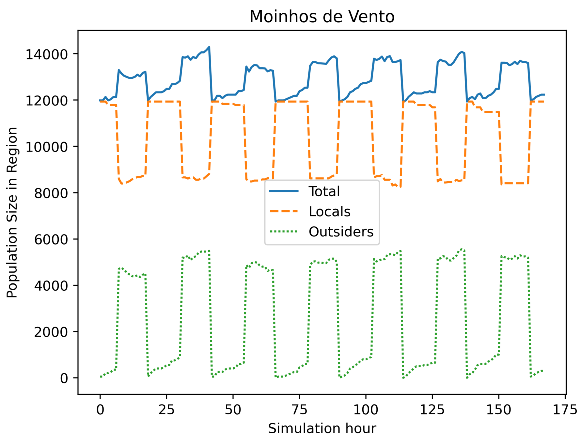
(c) Population over time in Moinhos de Vento with $s = 50$.



(d) Population over time in Centro with $s = 100$.



(e) Population over time in Floresta with $s = 100$.



(f) Population over time in Moinhos de Vento with $s = 100$.

Figure 6.29: Experiment H: Population over time in each of the analyzed neighborhoods. Combining the routine and Lévy distribution approaches to movement description. With movement restriction $r = 0.3$. The graphs compare Lévy movement scales of $s = 50$ and $s = 100$.

7. FINAL CONSIDERATIONS

This section presents our final considerations for the present work, which presented BioClouds, Legion and LODUS. This work aimed to study the little explored field of multilevel simulation models and methods that can make the transition dynamically to and from the microscopic and macroscopic levels of simulation.

BioClouds and Legion are space discretization macroscopic crowd simulation models, which abstract crowds into groups of individuals which move together, with similar goals and characteristics. While Bioclouds presented a multi-level visualization approach, using BioCrowds to visualize agents, if the simulation was intended to be microscopic. Legion presented a multi-level simulation approach which can make the transition between different levels of simulation dynamically, differently from hybrid models, like the model proposed by Xiong et al. [25] which have regions statically defined to behave with different levels of simulation. Legion also proposed a blobby visualization model for blob based crowd simulation models, using blobby models to generate a more organic occupation of area than BioClouds. Bioclouds [2] and Legion [1] resulted in papers published during the master's degree.

LODUS is a framework developed at the Virtual Humans Lab (VHLAB), it is composed of two main modules, a city-generation tool[1] and a mobility simulation model, developed in the present work. The LODUS mobility module is a population mobility model that uses crowd simulation based techniques to solve the human mobility task, particularly the groups as blobs of individuals from BioClouds and Legions. LODUS abstracts the population and blobs of individuals, described by characteristic histograms, defining how a population is distributed given the modeled characteristics. Environment is abstracted as Points-of-Interest (POI), containers of blobs, which represent places of particular interest for the simulation, e.g. houses, markets, schools, and regions which are groups of POIs.

The present work also proposed three use cases for LODUS, population mobility, public transportation and disease spread. A population mobility experiment, based on cities having their mobilities defined by both a rigid cyclical schedule and a noisy mobility behavior based on studies by Gonzalez et al. [8] and Brockmann et al. [4]. Public transportation could be modeled in LODUS by using mobility actions which propagate over several simulation frames. Disease spreading could be modeled combining the LODUS mobility approach and population description with a *SIR* simulation, using the population profiles to store variables for the *SIR* simulation and spread the disease using mobility. We performed qualitative assessment of experiment measuring the impact of mobility parameters for the population mobility. The experiments measured the population over time of highlighted neighborhoods with different emergent population behaviors.

The LODUS framework is a promising tool for modeling complex simulation scenarios with the set of environment generation tools and along with population mobility simulation. This could position LODUS as a useful tool for analyzing population mobility problems. This simulation

[1]Developed by Gabriel Fonseca Silva as part of his master's degree.

approach could be extended to explore: contagion modeling, public transportation designing tasks, improving the visualization population flow of a simulated city, modeling migration flows between countries or continental regions, modeling changes in population characteristic composition due to long term social policies (such as increasing schools in specific zones of a city or government expenditure in specific commercial zones).

Considering a given population mobility data for a given city, the visualization of daily population movements can be useful to guide decision making processes for this city. LODUS can be used as a population flow visualization tool by simulating a population, creating dynamic data for a given time frame based on static population descriptions. In a similar manner to public transportation, as described in earlier chapters, we can simulate long term and long range population mobility. This approach applies to long distance travels, such as air plane and maritime transportation, as well as long term population mobility, regardless of transportation mode, such as migration flow. This long term simulation is facilitated by LODUS environment abstraction, by using a different description for an environmental graph, transportation ratios and functions compatible with a simulation timestep measured in days or months. With our approach we can also explore a different subset of simulation models: population characteristics evolution over time, e.g., the change in education in a community with new schools. Instead of focusing on population mobility, this approach models longer spans of time and focuses on the changes of population composition instead of location. A LODUS style simulation would use functions to change the composition of blobs as they occupy regions with schools with a timestep representing a large timespan.

The applicability of LODUS to solve real-life problems is contingent to finding solution to some open issues. Validation of LODUS simulation data requires information of large population mobility which is not easily available, due to privacy concerns or cost. The visualization of LODUS generated data, the OD matrices and quantity over time graphs, is not easy to grasp or analyze, specially by non-experts. The models for public transportation is, at the moment, only theoretical, further investigations have to be made to validate an implemented case study. Regarding the disease spreading case, our research group is currently seeking to provide such integration.

BIBLIOGRAPHY

[1] Antonitsch, A.; Musse, S. R.; de Figueiredo, L. H. “Towards a legion of virtual humans: Steering behaviors and organic visualization”. In: SIBGRAPI Conference on Graphics, Patterns and Images, 2020, pp. 31–38.

[2] Antonitsch, A. D. S.; Schaffer, D. H. M.; Rockenbach, G. W.; Knob, P.; Musse, S. R. “BioClouds: A multi-level model to simulate and visualize large crowds”. In: Computer Graphics International Conference, 2019, pp. 15–27.

[3] Aurenhammer, F. “Power diagrams: Properties, algorithms and applications”, *SIAM Journal on Computing*, vol. 16–1, Aug 1987, pp. 78–96.

[4] Brockmann, D.; Hufnagel, L.; Geisel, T. “The scaling laws of human travel”, *Nature*, vol. 439–7075, Jan 2006, pp. 462–465.

[5] de Lima Bicho, A.; Rodrigues, R. A.; Musse, S. R.; Jung, C. R.; Paravisi, M.; Magalhães, L. P. “Simulating crowds based on a space colonization algorithm”, *Computers & Graphics*, vol. 36–2, Apr 2012, pp. 70–79.

[6] Durupınar, F.; Güdükbay, U.; Aman, A.; Badler, N. I. “Psychological parameters for crowd simulation: From audiences to mobs”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 22–9, Nov 2015, pp. 2145–2159.

[7] Favaretto, R. M.; Musse, S. R.; Costa, A. B. “Emotion, Personality and Cultural Aspects in Crowds: Towards a Geometrical Mind”. Springer-Verlag, 2019, 205p.

[8] Gonzalez, M. C.; Hidalgo, C. A.; Barabasi, A.-L. “Understanding individual human mobility patterns”, *Nature*, vol. 453–7196, Jun 2008, pp. 779–782.

[9] Harko, T.; Lobo, F. S.; Mak, M. “Exact analytical solutions of the Susceptible-Infected-Recovered (SIR) epidemic model and of the SIR model with equal death and birth rates”, *Applied Mathematics and Computation*, vol. 236, Jun 2014, pp. 184–194.

[10] Helbing, D. “A fluid dynamic model for the movement of pedestrians”, *Complex Systems*, vol. 6–5, May 1992, pp. 391–415.

[11] Helbing, D.; Molnar, P. “Social force model for pedestrian dynamics”, *Physical Review E*, vol. 51–5, May 1995, pp. 4282.

[12] Hernández-Orallo, E.; Armero-Martínez, A. “How human mobility models can help to deal with COVID-19”, *Electronics*, vol. 10–1, Jan 2021, pp. 33.

[13] Hughes, R. “The flow of human crowds”, *Annual Review of Fluid Mechanics*, vol. 35–1, Jan 2003, pp. 169–182.

[14] Jamshidi, S.; Baniasad, M.; Niyogi, D. "Global to USA county scale analysis of weather, urban density, mobility, homestay, and mask use on COVID-19", *International Journal of Environmental Research and Public Health*, vol. 17–21, Jan 2020, pp. 7847.

[15] Mandelbrot, B. B. "The fractal geometry of nature". W. H. Freeman and Company, 1982, 468p.

[16] Narain, R.; Golas, A.; Curtis, S.; Lin, M. C. "Aggregate dynamics for dense crowd simulation", *ACM Transactions on Graphics*, vol. 28–5, Dec 2009, pp. 1–8.

[17] Narang, S.; Best, A.; Curtis, S.; Manocha, D. "Generating pedestrian trajectories consistent with the fundamental diagram based on physiological and psychological factors", *PLOS ONE*, vol. 10–4, Apr 2015, pp. e0117856.

[18] Narang, S.; Best, A.; Manocha, D. "Interactive simulation of local interactions in dense crowds using elliptical agents", *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2017–3, Mar 2017, pp. 033403.

[19] Neto, A. B. F.; Pelachaud, C.; Musse, S. R. "Giving emotional contagion ability to virtual agents in crowds". In: International Conference on Intelligent Virtual Agents, 2017, pp. 63–72.

[20] Pelechano Gómez, N.; O'Brien, K.; Silverman, B. G.; Badler, N. "Crowd simulation incorporating agent psychological models, roles and communication". In: First International Workshop on Crowd Simulation, 2005, pp. 21–31.

[21] Silva, G. F.; Cassol, V.; Neto, A. B. F.; Antonitsch, A.; Schaffer, D.; Musse, S. R.; de Marsillac Linn, R. "LODUS: A multi-level framework for simulating environment and population - A contagion experiment on a pandemic world". In: IEEE International Smart Cities Conference, 2020, pp. 1–8.

[22] Thalmann, D.; Musse, S. R. "Crowd Simulation". Springer-Verlag, 2012, 296p.

[23] Treuille, A.; Cooper, S.; Popović, Z. "Continuum crowds", *ACM Transactions on Graphics*, vol. 25–3, Jul 2006, pp. 1160–1168.

[24] Van Den Berg, J.; Guy, S. J.; Lin, M.; Manocha, D. "Reciprocal n-Body Collision Avoidance". In: *Robotics Research*, Springer Berlin Heidelberg, 2011, pp. 3–19.

[25] Xiong, M.; Lees, M.; Cai, W.; Zhou, S.; Low, M. Y. H. "Hybrid modelling of crowd simulation", *Procedia Computer Science*, vol. 1–1, May 2010, pp. 57–65.

APPENDIX A – PUBLISHED PAPERS DURING THE MASTER'S DEGREE

- da Silva Antonitsch, André, Soraia Raupp Musse, and Luiz Henrique de Figueiredo. "Towards a Legion of Virtual Humans: Steering Behaviors and Organic Visualization." In: 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), 2020.
- Silva, Gabriel Fonseca, et al. "LODUS: A Multi-Level Framework for Simulating Environment and Population - A Contagion Experiment on a Pandemic World." In: IEEE International Smart Cities Conference (ISC2), 2020.
- Antonitsch, Andre da Silva, et al. "BioClouds: A multi-level model to simulate and visualize large crowds." In: Computer Graphics International Conference, 2019.
- Schaffer, Diogo, et al. "Towards Animating Virtual Humans in Flooded Environments." ACM SIGGRAPH Motion, Interaction and Games, 2020.