Machine Learning methods for optimization in Multi-Agent Decision Support System : application to Sign Placement for Tsunami Evacuation

Van-Minh Le

Université Paris Nord

Ecole doctorale Galilée

Machine Learning methods for optimization in Multi-Agent Decision Support System: application to Sign Placement for Tsunami Evacuation

Author:
Van-Minh LE

Supervisor:
Jean-Daniel ZUCKER
Yann CHEVALEYRE
HO Tuong Vinh

Laboratoire d'informatique de Paris Nord (LIPN)
Institut de Recherche pour le Développement (IRD)
Institut Francophone International (IFI)

December 13^{th}, 2016

Declaration of Authorship

I, Van-Minh LE, declare that this thesis titled, 'Machine Learning methods for optimization in Multi-Agent Decision Support System: application to Sign Placement for Tsunami Evacuation' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UNIVERSITÉ PARIS NORD

Abstract

Laboratoire d'informatique de Paris Nord (LIPN)
Institut Francophone International (IFI)
Institut de Recherche pour le Développement (IRD)

Doctor of Philosophy

Machine Learning methods for optimization in Multi-Agent Decision Support System: application to Sign Placement for Tsunami Evacuation

by Van-Minh LE

In recent years, whenever we talked about tsunami, we mentioned the terrible destruction and huge casualties (the tsunami from Indian Ocean in 2004 and the tsunami in Tohoku Japan 2011). The evacuation is the most effective solution to save people in this kind of disaster. Before the tsunami arrival, people should go to the high buildings (called vertical shelters) or high ground areas or zones far from the sea (called horizontal shelters). However, there are always the part of evacuees (e.g. the tourist) who lack information of the city map, we then focus on the solution to guide people in evacuation.

This report presents the approach of Efficient Optimization in a Multi-Agent Decision Support System: Application to Sign Placement for Tsunami Evacuation. More precisely, we study the approach to place signs and also evacuation maps in the city (at certain crossroads or junctions) to have as many people (call survivors) as possible reaching the shelters before tsunami arrival.

Acknowledgements

Firstly, I would like to thank Professor Jean-Daniel ZUCKER. He is the kindest professor that I have ever seen. I say that because I witnessed what he has done to help me and other students. During my phD, I received much help from another professor named Yann CHEVALEYRE. Personally, I think he is the best that I have ever worked with. His knowledge is beyond of my comprehension. I admire him. Finally, I would like to thank to 2 other men: Mr HO The Nhan and Mr HO Tuong Vinh. They gave me useful advices during my work.

Contents

List of Figures

List of Tables

Abbreviations

ABM	**A**gent-**B**ased **M**odeling
ABS	**A**gent-**B**ased **S**imulation
EBM	**E**quation-**B**ased **M**odeling
MAET	**M**inimize **A**verage **E**vacuation **T**ime
AET	**A**verage **E**vacuation **T**ime
ET	**E**vacuation **T**ime
SP	**S**ign **P**lacement
GA	**G**enetic **A**lgorithm
LR	**L**inear **R**regression
OR	**O**rdinal **R**regression

Constants

Average Speed of Pedestrian in Evacuation	$\mathcal{V}_{max}$	=	$1.5\mathrm{ms}^{-\mathrm{S}}$
Discretization coefficient	α	=	5.0
Number of **R**epeating **T**imes of **S**imulation	N_{RTS}	=	100

Symbols

α	Discretization coefficient
$a_{ij} \in \{0, 1\}$	indicated whether a sign was placed on arc $\{i, j\}$
A	Arc set
C	Cost matrix
c_{ij}	The time (in seconds) requires an agent to move from vertex i to vertex j
K	Maximum number of signs
L	Maximum number of Local Evacuation Map
M	number of agents
μ_i	Initial distribution of people (or evacuees) at vertex i
N_{RTS}	**N**umber of **R**epeating **T**imes of **S**imulation
$\mathcal{P}$	**P**opulation in genetic algorithm
P	Transition matrix
p_{ij}	Probability an agent to turn from vertex i toward vertex j
$Pr[t > T\|i]$	Probability for an agent to start at vertex i to reach a shelter in more than T seconds
q_i	the Average Evacuation Time for pedestrians beginning at vertex i
$q_{i,k}$	Probability for an agent starting at vertex i to reach a shelter in time $t > k\alpha$
S	Sign position set
SP	**S**ign **P**lacement
SR	**S**urvival **R**ate
$\mathcal{V}_{max}$	Maximum speed of pedestrian
V	Vertex set
X	Shelter set

Dedicated to my mother, the one that I love the most.

Chapter 1

Introduction

In this chapter, we present our problem as clearly as possible. Generally, the problem is how to build a system that can automatically propose an optimal guiding system in order to lead people in the evacuation from tsunami disaster.

1.1 Tsunami

First, what is tsunami? According to the organization National Oceanic and Atmospheric Administration, tsunami is "a Japanese term derived from the characters **tsu** meaning harbor and **nami** meaning wave". Tsunami is "now generally accepted by the international scientific community to describe a series of traveling waves in water produced by the displacement of the sea floor associated with submarine earthquakes, volcanic eruptions, or landslides."

The most important problem is that tsunamis become one of the most dangerous natural disaster for coastal regions. In fact, the tsunami from Indian Ocean in 2004 and the tsunami in Tohoku Japan 2011 (in SUPPASRI et al. [2012]) which took thousands of human lives are the examples of these destructive disasters.

How to deal with tsunami? In order to deal with tsunami, most studies focus on 3 directions:

1. Warning guidance: The main purpose of this direction is how to forecast the tsunami (presented in TITOV et al. [2005]). Researchers focus on Deploying Tsunami Detection Buoys and Improving Seismic Networks.

2. Hazard assessment: Most studies aim at Producing Tsunami Inundation Maps in order to evaluate how dangerous an area is when tsunami arrives (introduced in GONZALEZ et al. [2009]).

3. Mitigation: In this direction, most studies focus on how to reduce the damage of the tsunami such as: the vegetation bioshields to reduce the tsunami power (described in TANAKA [2009],DANIELSEN et al. [2005]) or reduce the victim of tsunami (e.g. the research on people evacuation in TANIOKA et al. [2012], SAITO and KAGAMI [2004], GOTO et al. [2012], SCHEER et al. [2011b])

In the condition of our work, we totally focus on the evacuation, a narrow domain of mitigation direction. In fact, our research aims to bring application to the developing countries (e.g. Vietnam or Indonesia) who do not have a high technology base or a huge investment of money. For these countries, the evacuation becomes the most effective solution because it can help reducing casualties (or more accurately saving people).

1.2 Evacuation in case of tsunami

Evacuation (or emergency evacuation) is an urgent movement of people away from the dangerous zones caused by disaster (in this case, disaster is tsunami, the most dangerous natural disaster). An evacuation is normally organized into the following steps:

1. Detection and decision: It is true that a tsunami happens after an earthquake in the ocean. An scientific department who is responsible for earth and geophysics problems might have the information about the earthquake. They will decide if there would be a tsunami.

2. Alarm: If the tsunami is predicted to happen, an alert information is sent to local authorities of the cities or regions where would be suffered by the disaster. The local authority then declares the evacuation alarm, and spread it through all communication means.

3. Movement to an area of refuge or an assembly station: From the point of view of the local authority, people should move to safe places before tsunami arrival. The safe places in this case are called shelters. There are two types of shelters: vertical shelters represent buildings; horizontal shelters represent high ground areas or the zones far from the sea. The figure 1.1 presents logos horizontal and vertical shelters (figure from SCHEER et al. [2011a]). The figure 1.2 shows an example of a shelter in the simulation.

FIGURE 1.1: Logos of horizontal shelters and vertical shelters

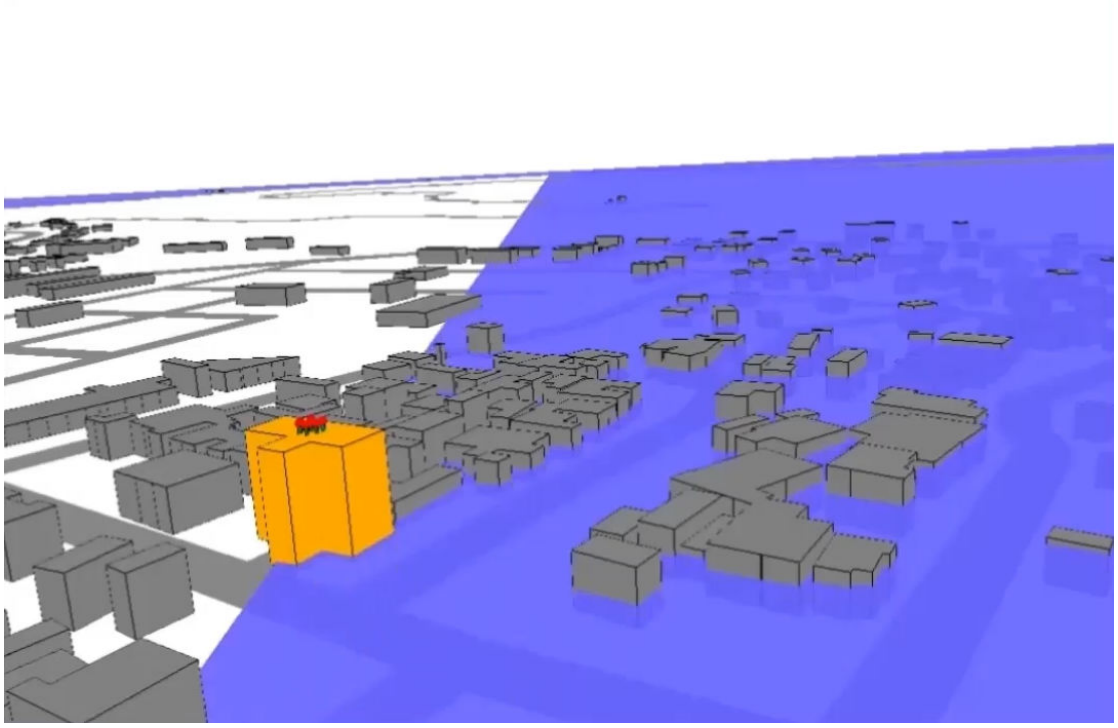


FIGURE 1.2: An example of a vertical shelter in our simulation

The main purpose of one evacuation is to have as many people as possible reaching shelters. However, in the evacuation there are some parts of population lacking information about the evacuation. There is some part of tourists who do not know where the shelters are. They do not know either where to go in case of tsunami. There is even part of citizens who do not know where the shelters are. Then, providing information for evacuees plays a very important role in the evacuation.

FIGURE 1.3: An example of an sign for tsunami evacuation

1.3 Placing signs to help people to evacuate

Guiding sign placement is one of the most effective way to provide information to evacuees. A sign (figure 1.3) is a panel situating at a corner or a junction in the city in which there is a logo representing tsunami wave with the direction and some optional information in local language (e.g. The figure 1.4 presents a sign in a city on which there are the sign direction and information about evacuation route). The figure 1.4 shows an example of a sign in our simulation. Then, a guiding sign placement (or sign placement for short) is a set of signs. Each sign has a specific location and specific direction.

During the evacuation, when people see a sign, they usually follow its direction. When they see a place with the shelter symbol (figure 1.1), they stay in this place for further instructions. The most important purpose of the evacuation is to have as many people as possible reaching shelters before tsunami arrival.

1.3.1 Survival rate

Survivors are the people who reach the shelters before tsunami arrival. Survival rate (**SR**) is the percentage of survivors over the population. In our work, SR is the most important indicator which is used to evaluate how good a sign placement is.

FIGURE 1.4: An example of a sign placed in the city

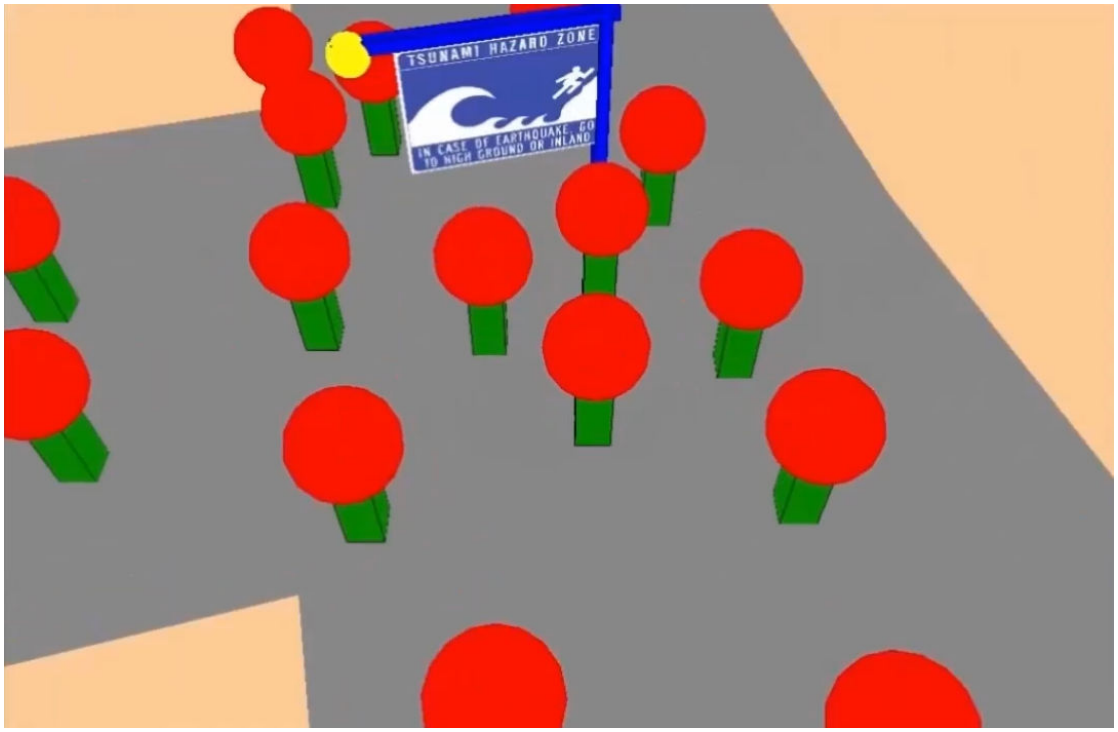


FIGURE 1.5: An example of a sign in our simulation

1.3.2 Problem of optimizing sign placements

As we mention above, the purpose of the evacuation is to save as many people as possible. Therefore, the objective function of the problem of optimizing sign placements is the maximized survival rate.

In the evacuation, there are always some people who do not know where to evacuate. Then, the number of survivors of the evacuation depends on how many signs are well placed. Hence, the more signs are placed, the more people are saved. If we have unlimited signs to put at every corner of the city (or every crossroads), everyone can easily evacuate. However, we have limited budget to place a limited number of signs (**K**).

Thus, our problem of optimizing sign placements is how to place a limited number of **K** signs in a city in order to maximize the survival rate (**SR**).

1.4 Content of this document

We first begin with the main problem (or the ultimate goal of our work), the problem of optimizing sign placements in the evacuation in order to maximize the number of survivors (or survival rate). From the most appropriate approach, we propose solutions to solve the main problem. However, the solutions might have some issues which become sub problems. We then present the solutions for these sub problems. Thus, we repeat these actions until all the problems have relatively acceptable solutions.

This document is organized as follow:

Chapter 1: we present the purpose of our study and the problem of sign placement in evacuation

Chapter 2: we present how to build a simple Agent-Based model of pedestrian evacuation in order to evaluate how good a sign placement is.

Chapter 3: we introduce the optimizing method of sign placement in which the objective function is evaluated by an Agent-Based simulation.

Chapter 4: we make the optimization method tractable. In fact, our experiments show that the Agent-Based simulation (proposed in chapter 3) is time-consuming, which makes the optimization approach far from feasibility. What we present in this chapter is a very fast surrogate model which replaces the Agent-Based model.

Chapter 5: we propose another aspect of the optimization problem. From this aspect, the problem becomes decomposable into two parts: optimization of sign positions and optimization of sign directions. We also propose the solution in order to improve the speed of optimization phase.

Chapter 6: we propose an approach to deal with the more complex model, in this case, the crowd model.

Chapter 7: we propose another approach to evaluate a sign placement with aspect of machine learning. In this case, we predict the survival rate of a sign placement by regression.

Chapter 8: we propose another guiding panel, called the local evacuation map, which considerately increases the survivors.

Chapter 9: we conclude what we have done and propose perspectives of our work.

Chapter 2

Simulating the Evacuation of Crowds

In the first chapter, we present our problem of optimizing sign placements. The main goal is how to find the optimal sign placement (consisting of **K** signs) in order to maximize the survival rate (**SR**). The first step to solve the optimization problem is to compute the objective function, which mean that we must evaluate how many survivors that a sign placement can save.

In this chapter, we present the way to build an evacuation simulation in order to evaluate how good a sign placement is. In fact, there are plenty of methods to solve an optimization problem, but all of them need an objective function. Moreover, no matter how the optimal solution is found, it must be eventually re-evaluated by a simulation which is closed to the reality. These are the reasons why we need evacuation simulation this work. Our subject is evacuation in case of tsunami but we also make a survey on other types of evacuation.

This chapter is organize as follow: the second section is the survey on Computer Models of Pedestrian Evacuation; in the third section, we present the survey on Agent-Based modeling approach; in the forth section, we propose our Agent-Based model of evacuation; and then, in the fifth section, we present the implementation and evaluation.

2.1 Introduction

In order to evaluate a sign placement, we need a model of pedestrian behavior, on which the simulation will be built. In order to choose a specific model which is the

most appropriate to our problem, we make a short survey on computer models of the pedestrian evacuation.

2.2 A Survey on Computer Models of Pedestrian Evacuation

2.2.1 Equation-Based Modeling vs Agent-Based Modeling

With respect to how to model an evacuation, Equation-Based Modeling (**EBM**) (in HENDERSON and NELSON [2006]) and Agent-Based Modeling (**ABM**) (in MACAL and NORTH [2011]) are the two emerging approaches to simulate the system by constructing a model and running it on computer. The differences between these approaches are how each of them forms the model and how that model is executed. In Equation-Based Modeling, the system is modeled by a set of equations and is executed by solving these equations. On other hand, in Agent-Based Modeling (in SHIFLET and SHIFLET [2014]), a system is modeled by a set of agents who interact via their behaviors, and is executed by emulating these behaviors.

While Equation-Based Modeling and Agent-Based Modeling compete to each other, the two approaches can be combined together in some cases. In PARUNAK et al. [1998], authors summarized similarities and differences of two approaches via a case study of supply network, then presented the criteria for selecting one or another approach. In NGUYEN et al. [2012c], authors proposed a hybrid model using either of these methods of modeling for simulating a crowd.

We agree that each approach has its own pros and cons. In our concrete work, we focus on building a simulation of evacuation in order to evaluate how good a sign placement is. Then, in this chapter we then begin to explore the basic model of evacuation, the agent-based pedestrian simulation. The equation-based model of pedestrian is also discussed later in chapter 4.

2.2.2 Indoor evacuation vs outdoor evacuation

Regarding Agent-Based Modeling, a model usually consists of an environment and a set of agents. With respect to evacuation environment, most of evacuation models focus on two types: indoor evacuation and outdoor evacuation. Indoor evacuation is the evacuation inside the building in which the main purpose of evacuees is how to get out of that building as soon as possible (in ROGSCH et al. [2014]). The popular example

for indoor evacuation is the evacuation in case of fire in the building (studies in LE et al. [2010], NGUYEN et al. [2012b], RONCHI and NILSSON [2013]). Outdoor evacuation is the evacuation happening in plain air where the most important goal of evacuees is to find a shelter (a building or a safe area) to stay there (in GOTO et al. [2012], LIU et al. [2010], MAS et al. [2012]).

While both types of evacuations are the movement of people from dangerous places to safe places, there are also differences in the nature of each evacuation. The very simple difference between these types of evacuation is where the evacuees should go. While people want to get out of facilities in indoor evacuations, they would rather enter the safe building in outdoor evacuation. Beside the target destination of evacuees, the environment of evacuation decides which are the important factors in each case of these evacuations, which means that some factors in outdoor evacuation are very important but they are less important in indoor evacuation and vice versa.

In indoor evacuation, how evacuees move is more important than how they choose the destination. Regarding the density of people, the environment for indoor evacuation is much more narrow than that of outdoor one. Then, the most important focus is the collision of evacuees and the factors effecting that collision. In SMITH and BROKOW [2008], authors focused on analyzing the size of evacuees and the width of corridors. In HELBING and MOLNAR [1995], authors proposed the social forces that affect evacuees and also the collision happing during the evacuation. In MUSSE and THALMANN [1997], authors presented the relationship among the evacuees which make them form into groups. The emotional factors (e.g the fear) also effect the evacuees' behaviors (introduced in LE et al. [2010], TSAI et al. [2011]). The fact is that the evacuation environment (in this case, the facility) is small enough, so the evacuees can easily find the way out or at least they can take the path to the entrance where they initially enter the facility.

In outdoor evacuation, the way that evacuees choose the path to escape is much more important than the way to move on the street. In fact, the street is wide enough for pedestrians to move on which means that even when the street is extremely crowded, it is not blocked. This is in sharp contrast to the indoor evacuation that the corridor sometimes is blocked when it is overcrowded and the evacuees might push each other to cause damage or even casualty. With respect to evacuation environment, the city is large enough for a person to need guiding information. In MINAMOTO et al. [2008], LIU et al. [2008], NGUYEN et al. [2012d], the authors introduced models of evacuation in which each evacuee was represented by a point moving along the lines representing the roads in the city.

2.2.3 Arguments to choose Agent-Based Simulation for outdoor evacuation

As we mention in the beginning of this chapter, the purpose of this chapter is to evaluate a sign placement by using a simulation of evacuation in case of tsunami. It is no doubt that the evacuation environment here is the city. We then decided to begin with outdoor evacuation. Regarding modeling method, we began with Agent-Based modeling to build the first simulation (presented in the next section). The Equation-Based modeling and the hybrid approach will be introduced in the next chapters.

2.3 Agent-Based Simulation of evacuation

As we mention in previous section, we begin to study Agent-Based Model of outdoor evacuation. Before building our model, we make a survey on how to model this kind of evacuation.

2.3.1 Survey on how to build a Agent-Based Model of outdoor evacuation

2.3.1.1 Approaches to model the environment

First, the traditional approach for modeling pedestrian behaviors in evacuation is modeling pedestrian movement on a grid. Each step of simulation, a pedestrian chooses one of the neighbor cells for his next location. Although this approach has received much success in simulating the evacuation from the building where the map is really small (the modeling grids proposed in SMITH and BROKOW [2008], CHRISTENSEN [2008]), we argue that it would not be a good choice for the evacuation of tsunami because the evacuation map of the real city is much larger than that of a building.

Another approach that is widely used to model the outdoor evacuation is the graph. The crossroads in the city are modeled as the vertices and the roads connecting them are represented as the arcs. Each agent (representing an evacuee) would move along the arc from a vertex to another vertex. The next destination vertex of an agent is one of the neighbor vertices of the current location of that agent. While this approach seems to be simple to model a pedestrian evacuation (introduced LIU et al. [2008]), it also provides an easy way to extend the model into more complex one. An example of extensibility of graph was presented in NGUYEN et al. [2012d], in which authors proposed the model

which distinguishes the knowledge levels of agents: some know all the map, the others do not.

A special type of graph proposed to model pedestrian behaviors in tsunami evacuation is the petri-net (in MINAMOTO et al. [2008]). In this approach, the pedestrian movement is modeled as the transitions. With this approach, modeler may add the rules inciting agents to go towards the less crowded road.

While petri-net approach provided the way to reduce overcrowded row, it opened a question how an evacuee knew which road is crowded. In fact, since agents normally "observe" what is happening locally, they can make their decisions based on the local perception of the environment. That is why we do not use this approach in our work. Thus, we propose to use graph to model the outdoor evacuation environment. In case we use graph to model environment, a agent's movement would be modeled as a point moving on the lines representing the roads, which opens another question how to distinguish the speed of evacuees in crowded roads with that speed in uncrowded roads. In order to bring the simulation close to the reality (which makes the evaluation of sign placement more reliable), we have to describe the traffic jam in the model.

2.3.1.2 Approaches to model traffic jam

It is no doubt that the speed of evacuees in the crowded roads is slower than that in uncrowded one. The model of the traffic jam in outdoor evacuation becomes the model of relation between density and movement speed. In FANG et al. [2003], authors presented a general relation between high density and movement velocity called logarithmic relationship. Since this is a general approach, we find that it is far from our problem.

Another proposition which is very close to evacuation problem was introduced in GOTO et al. [2012]. In this proposition, the speed of a pedestrian was computed by a formula between the maximum pedestrian speed and the density in front of that pedestrian. Speed decreased with increasing density. In our work, we decided to use the formula in GOTO et al. [2012] to describe the relationship between the evacuees' speed and crowded density because we found that it is close to the evacuation and then fit to our work. More details of this relationship are written in the next section.

2.3.1.3 Approaches to model agent behaviors

The heart of an Agent-Based model is agent's behaviors. In order to build an Agent-Based simulation, we need to consult the related works on how to describe evacuees' behaviors in evacuation.

The simplest approach for agents' behaviors is reactive behaviors. It is simply that an agent (representing an evacuee) react to what it perceives. In fact, the reactive behavior is simply like an action in a specific condition (described in METOYER and HODGINS [2004], PELECHANO and N.I. [2006], SHAO and TERZOPOULOS [2005]). An example in our case of sign placement in evacuation is that if an agent perceives a sign, it follows the direction of that sign. Otherwise, that agent evacuates randomly.

Another approach which is more convincing for pedestrian model is cognitive behaviors, for the agents represent the people whose behaviors are very complex. Most of cognitive behaviors are based on psychologic theory (e.g the studies in VORST [2010]). Concretely in FRIDMAN and KAMINKA [2007], authors modeled the behaviors which were based on Festinger's theory (described in FESTINGER [1957]). We agree that the more realist behaviors are, the more reliable results are produced. However, the complex behaviors cost much computational resource, including memory resource and computation time.

Crowd-based behaviors are the actions that groups of agents tend to form. Most of studies on emergency evacuation aim at these behaviors. While the facts showed that people evacuate in group in order to share their help to each other, some studies analyzes the negative effect of the crowd which reduces the speed of evacuees (in CHOW [2007]). For the outdoor evacuation, the groups also share information about the routes and the shelters for its members, which is very useful for outdoor evacuees. In fact, while some studies claim that cognitive behaviors make agents to form crowds (in BELTAIEF et al. [2011], PELECHANO et al. [2005]), others propose other reactive simple models in which agents could create crowds (details in MUSSE and THALMANN [1997], QIU and HU [2010], JI and GAO [2007]). Our proposed model of crowd is introduced in chapter 6.

2.4 Agent-based models of pedestrian evacuation

As we explain in the previous section, we have to build a simulation of an outdoor evacuation because the evacuation from tsunami happens in the plain air where every evacuee needs to reach a high building or a high-ground area. We choose the agent-based approach to build the model of this type of simulation because it addresses direct the problem. In our first model, the main problem is the fact that some people do not know where the shelters are nor how to get there. If they see a guiding sign, they follow it. Otherwise, they will move randomly until they acquire other information. Then, we focus on the these people whose behaviors are quite simple. Therefore, we decided to study first the simple model of an outdoor evacuation in which agents' behaviors are reactive. This section describes details of this models.

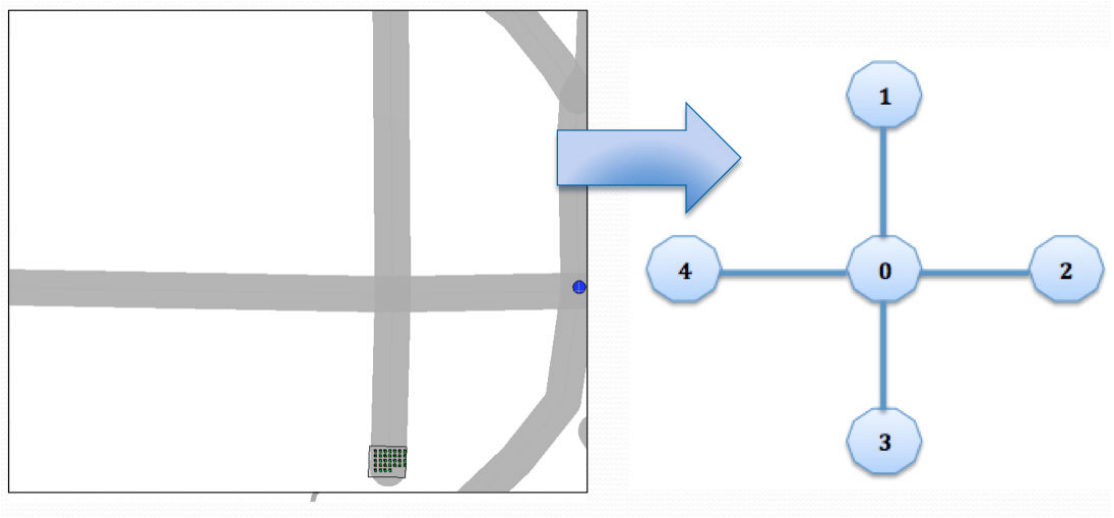


FIGURE 2.1: Example of modeling map into graph

2.4.1 Simple reactive behaviors of pedestrian in evacuation

2.4.1.1 Environment description

First we model the environment, in this case the city map, into a graph. The figure 2.4 presents the map of the city where we simulate the pedestrian evacuation. The data representing the city map is formatted in GIS provided by Institute of GeoPhysics of Vietnam.

In this case, each crossroad or junction is represented by a vertex. The roads between these crossroad are modeled as the arcs. The figure 2.1 shows an example of crossroad which is modeled as a graph of 5 vertices and 4 edges (each edge consists of 2 arcs).

Evacuation time is the limited time for evacuees to move to the shelters. According to scenario of tsunami, length of evacuation time is predefined before the experimentation of any model. For the example of a scenario described in VU and NGUYEN [2008], the tsunami is supposed to form from an earthquake on Manila Trench, the first wave would arrive Vietnamese coast in 2 hours; the evacuees in this case could have 2 hours to evacuate; Then, the evacuation time would be set as 2 hours (or 7200 seconds). Another scenario in NGUYEN et al. [2012a] describes that an earthquake is supposed to happen very near the coast of Vietnam and the local spends time to decide to announce the evacuation alert; the evacuation time for people to evacuate would be very short as 15 minutes (equal to 900 seconds); the evacuation time for this case is then 900 seconds.

The shelters in this case are the building and high ground area. The vertex which is nearest to a shelter is considered as shelter vertex. The figure 2.2 shows an example of shelters. In this example, there is 2 high building near the vertex A and vertex H. These

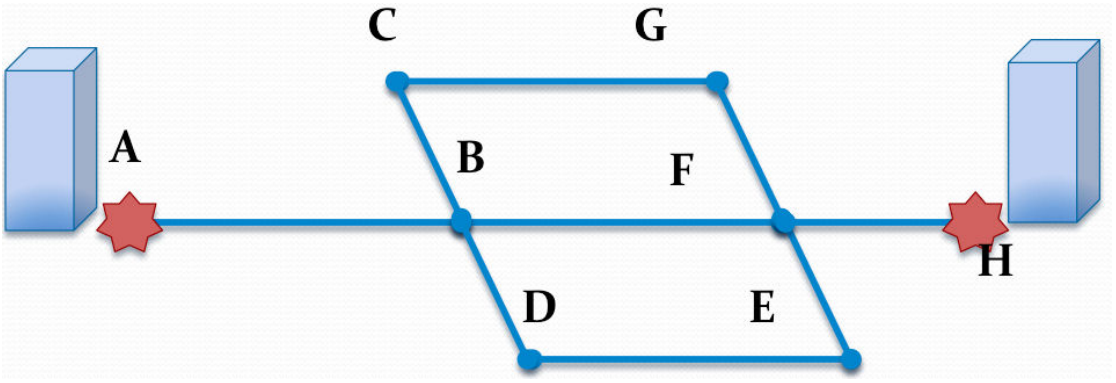


FIGURE 2.2: Example of modeling shelters

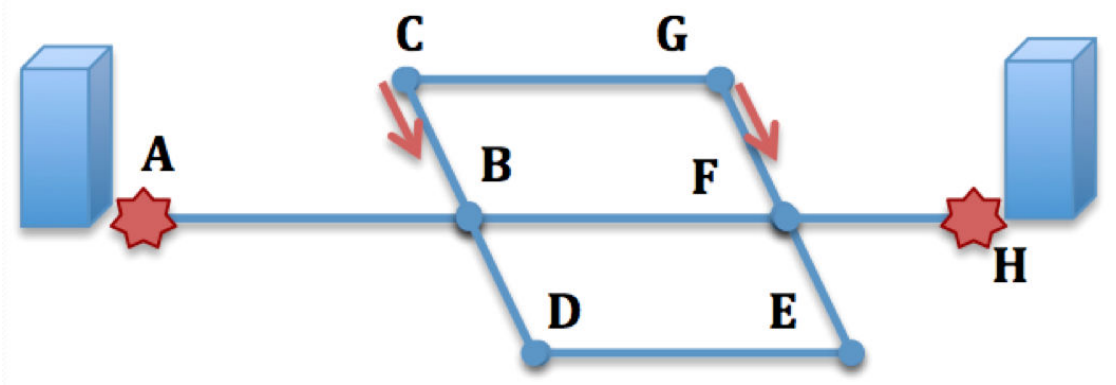


FIGURE 2.3: Example of modeling shelters

2 vertices are considered as shelter vertices. Once agents arrive one of the 2 vertices (before tsunami arrival), they are considered survivors.

2.4.1.2 Input parameter and outcome result

According to the model of environment, the input parameter of this model is a list of K guiding signs. A guiding sign is defined by a pair of vertices. The first vertex represents the location of the sign while the second one represents direction. The direction of a sign in this case is one of the neighbors of the vertex representing the location. The figure 2.3 shows an example of a sign placement which contains 2 signs: one sign locating at vertex C pointing to vertex B, another locating at G pointing at F. Then, in the simulation, the sign placement can be described as follow:

$$SP = \{[C, B], [G, F]\}$$

Output of the simulation of this model is the percentage of survivor (or survival rate). The agent who arrives one of these vertices within the evacuation time is considered survivor.

2.4.1.3 Agent description

Since this is the model of reactive agents, the agents in this case are very simple. We model an agent as a point. The way that an agent performs its evacuation is modeled as the movement of a point from a vertex to another vertex along the arc. The speed of agents in this case is simply set as Average Speed of Pedestrian in Evacuation (avpe = 1.5 m/s).

In this model, there is only one type of agents representing the tourist who does not know the map. The behaviors of this type of agent are simply composed of 2 actions:

1. Moving: if a agent is on an arc, it moves to its next destination.
2. Choosing next destination: if a agent is at a vertex, it chooses next destination as follow:
   (a) If there is a guiding sign at that vertex, the agent uses the second vertex of the pair describing sign as the next destination.
   (b) If there is no guiding sign at that vertex, the agent randomly takes one of the neighbor vertices as the next destination.

2.4.2 Implementation

In this case, we build our simplest model in GAMA (version released by DROGOUL et al. [2013]). This model simulated the evacuation of pedestrians in the case of tsunami in Danang city in Vietnam. In our simulation we used the real map of this city (figure 2.4) and simulated 10000 pedestrians with the initial uniform distribution on over the junctions of the map.

We focus on the scenario of evacuation in this simulation is that: there would be an earthquake which of 8.0 on Richter scale from Manila Trench which could cause the tsunami high up to 4 meters. The tsunami propagation time from the source region to Vietnamese coastal area were estimated about 2 hours (described in VU and NGUYEN [2008]). For the normal case, the evacuation is set as one hour. The evacuation time and the number of pedestrians might vary in different cases of experimentation. In the next chapter, we analyze the worst scenario of disaster, the evacuation time is set as 15 minutes according to this scenario.

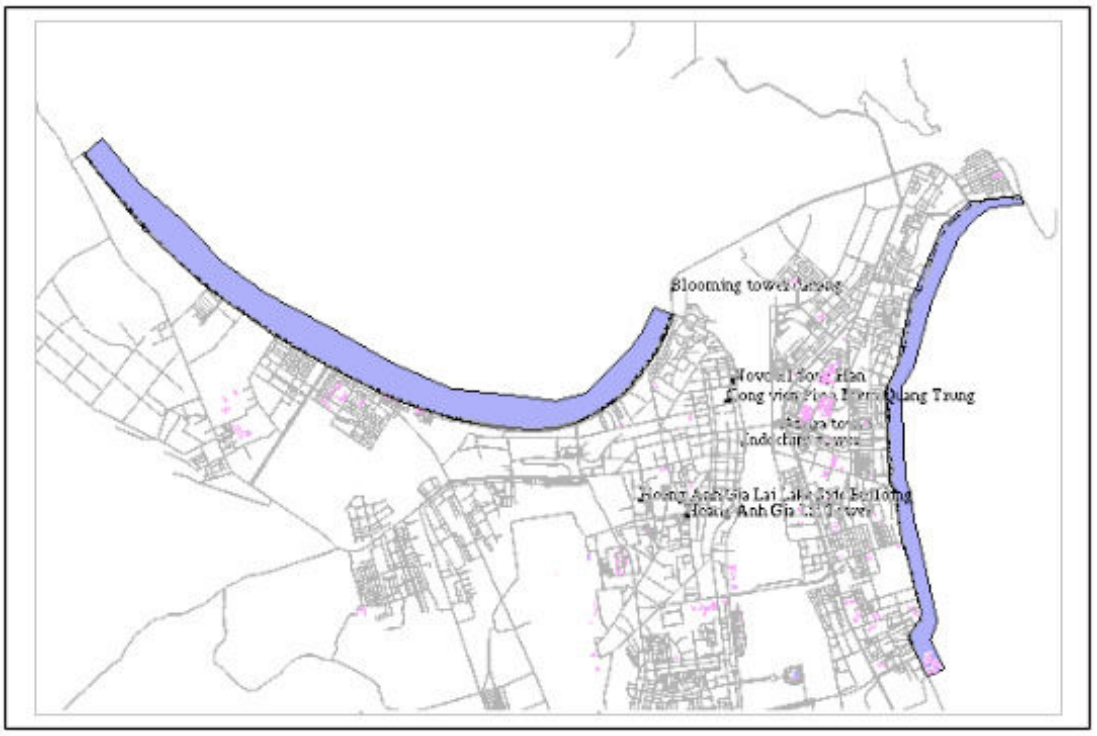


FIGURE 2.4: Map of Danang city, Vietnam

2.5 Experimentation and discussion

This is the experimentation for the first of our models. The most important purpose of this experimentation is how valid a model is. Even with a simple model, there are plenty of impacts which would make the output result far from the reality.

2.5.1 Discrete time in simulation vs continuous time in real world

The first impact is how real the simulation is with respect of time. A clear example is that: an agent will take 500 steps to move from its initial location to the nearest shelter. Knowing that computer must calculate and move that agent from its location to the next location and the iteration of this calculation is called a step, the movement here would cost the computer 500 iterations of calculation. The question here is how long it takes a person to move the same distance in the real world, it is 500 seconds or 500 minutes or some other values.

In this case, we want to calibrate the simulation so that 1 step in the simulation (run by computer) is approximately equal to 1 second in the real world. This issue might be complicated in many simulation plate-form but in GAMA (in DROGOUL et al. [2013])this calibration becomes tractable. Thanks to ability to convert the length unit

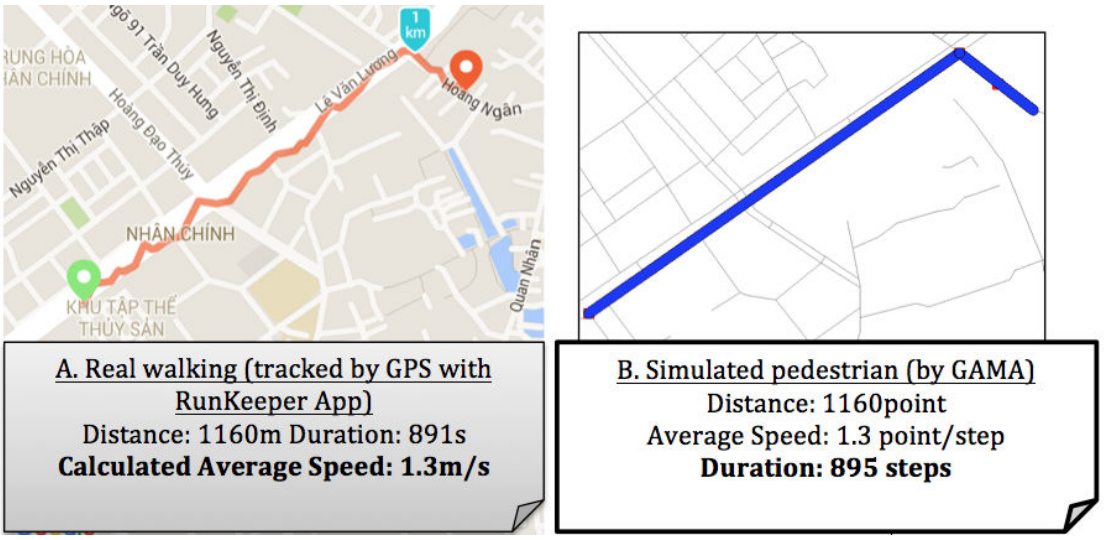


FIGURE 2.5: Real pedestrian and simulated pedestrian

of GAMA, we can use the GIS map under the metric unit. We deploy our experiment by following steps (detail in figure 2.5):

1. Real walking person: We let one man walk a long a street.
2. Tracking real movement: The movement is tracked by GPS using a smart phone application. In this case we use Android RunKeeper. We observe the walking time and distance, which means that we want to know how long the distance is and how much time it takes the man to finish this distance.
   (a) Observed distance: In this case, the distance is 1160 meters.
   (b) Observed walking time: The time is 891 seconds.
3. Real average walking speed: We divide the distance by the walking time to get the real average walking speed. In this case, the value of this speed is 1.3m/s
4. Simulated walking person: We make an Agent-Based simulation of 1 agent representing only 1 pedestrian walking in the same environment as the real walking person.
   (a) Organize the GIS data: The map is provided by OpenStreetMap under the GIS format. In this case, we reorganize these GIS data so that the unit of length is in meter.
   (b) Simulated road is modeled as the same road in the real world (described in the step 2.a.). The distance of the simulated road is the same as that of walking road. In this case, the length of simulated road is 1160 units. By observing the simulated road in the part B of figure 2.5, the road is a little

longer than that of the real walking road but the lengths of both roads are the same length because of the relative incorrect GPS tracking which makes the real walking a little zigzag.

(c) Set agent speed: The speed of agent is set the same value as the real average walking speed. In this case, agent speed is set 1.3 units/step. Knowing that 1 length unit in simulation is approximately equal to 1 meter in the real world, we expect that 1 step in simulation corresponds 1 second in the real world.

5. Observed number of steps: Once the agent finished the movement (to the end of the simulated road), the number of steps is outcome. In this case, we repeat this simulation 100 times and the outcome value is 895 steps.

We do the same experimentation in the two other different roads, the outcome simulated steps is relatively equal to the observed time of the real movement, which leads us to the conclusion: We can calibrate the model so that a step in the simulation is approximately equal to a second in the real world. In general, it is acceptable that number of steps in simulation is equal to the same number of seconds in the reality with respecting to the evacuation time of pedestrian.

This case of experiment along with its conclusion is very important for our next experimentation and also next chapters, because from this moment we can assume that the simulation time (in this case is the step) is equal to one second in the reality.

2.5.2 Number of agents vs execution time

In this experimental setting, we introduce the factor that impacts most of Agent-Based Models, the relationship between the number of agents and execution time. Since behaviors of every single agent in Agent-Based model must be calculated and simulated (by computer), it is quite reasonable that the more agents, the slower simulation becomes.

2.5.2.1 Description of experiment

In this experiment, we run our model with the full map of the city (described in figure 2.4). The map is modeled as a graph which has 3452 vertices and 5226 edges (approximately 10452 arcs). The population of the city (in the year when the experiment are carried out) is approximate 1000000 people (one million people).

The input sign placement of 100 signs is set randomly. This sign placement is fixed for every run of the simulation. The execution time is set 7200 seconds corresponding two hours of spreading of the first wave of tsunami.

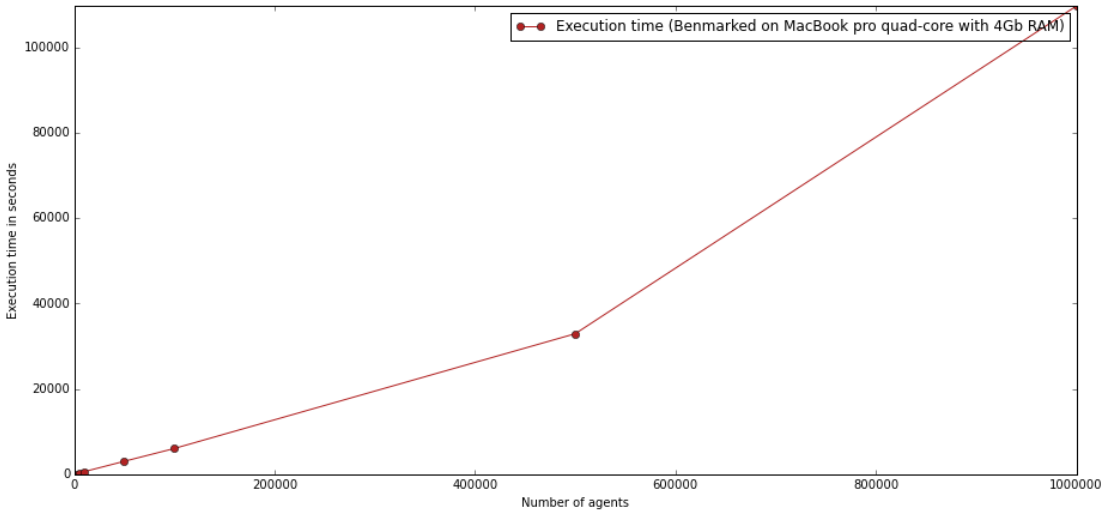


FIGURE 2.6: Execution time depends on number of agents

Regarding number of agent, it is required that we need a simulation of 1000000 agents (one million agents). In order to illustrate the relationship between the number of agents and execution time, we run the simulation with the various number of agents. The number varies from 1000 (thousand) to 1000000 (one million). All these cases of test are benchmarked on the same computer (MaBook pro quad-core with 4Gb RAM).

2.5.2.2 Results and discussion

The figure 2.6 shows the results of the experiment. As we expected, the execution time grows along with the augmentation of the number of agents. From observation of experiment's outcomes, we notice that:

1. It takes about 70 seconds to run the simulation of 1000 agents.
2. The simulation of 10000 (ten thousand) agents requires around 593 seconds (around 10 minutes)
3. The simulation of 1000000 (one million) agents needs about 109671 seconds (more than 30 hours)

The most important conclusion for this experiment is that the Agent-Based Simulation needs too much time to run a model of real population. For the simulation of real population, it takes more than 30 hours to finish only one evaluation of sign placement. This is infeasible in the current condition of our computer power.

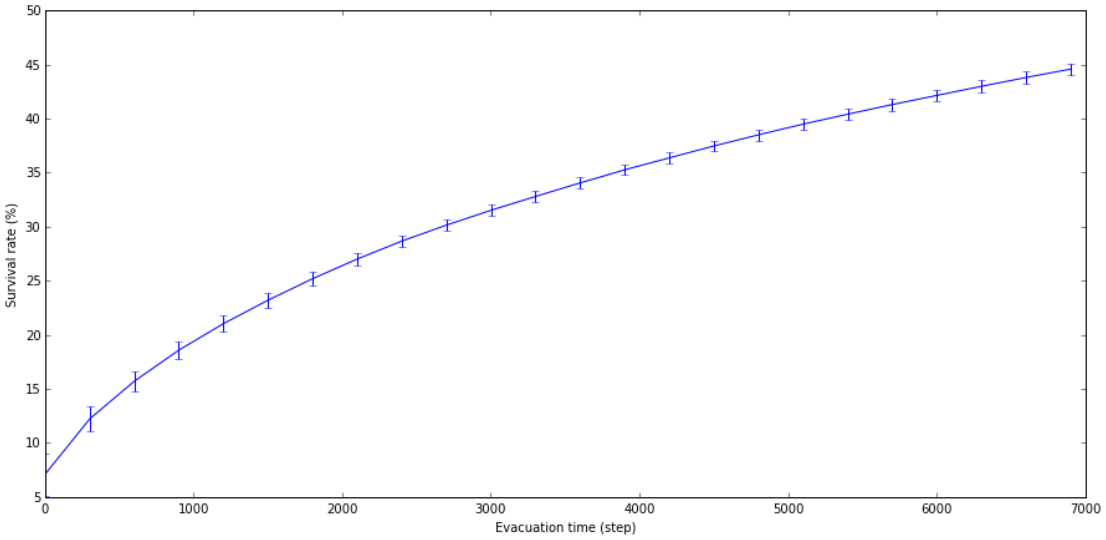


FIGURE 2.7: Stochastic results from simulation

2.5.3 Repeatedly running times vs reliability of result

In two previous experiments, we mention only the execution time and evacuation time. In this experimentation, we examine the survival rate (the most important purpose of the simulation).

As we present in the behaviors' description, an agent turns randomly if it perceives no guiding sign. Then, an agent makes different choices even if it is in the same situation. The consequence is that the outcome (in this case, survival rate) becomes stochastic, which means that each running time the simulation gives different survival rates with the same input of sign placement.

For this experiment, we reuse the model and parameter as the previous experiment with the same map, the same sign placement, the same evacuation time. The only difference of two experiments is the number of agents. In this case, the number of agents is 10000 agents corresponding ten thousand people. For each step of simulation, we log the number of survivors, and then calculate the survival rate. We repeat running the simulation 100 times.

The most important information of the result is the variance. The survival rates in 100 times are almost different. From the result (figure 2.7), we go to the conclusion that even with the same input parameter, the outcome is stochastic. This experiment opens an important question that how many repeating times is acceptable for a reliable outcome. In fact, the authors in MAURER and PONTIL [2009] have proposed formulas which indicate how many sample must be evaluated to get a reliable outcome.

This experiment result once again emphasizes that the Agent-Based simulation needs much time to run, which is called time-consuming. The cause of this time-consuming problem is assumed to be the nature of Agent-Based approach.

2.6 Conclusion

2.6.1 Agent-Based Simulation for evaluation of sign placement

In this chapter, we present how to build a simple Agent-Based simulation of pedestrian evacuation in case of tsunami. The most important purpose of this Agent-Based simulation is to evaluate how good the simulation is. We can have plenty of methods to optimize the sign placement, but in the end we must need a simulation to reverify the optimal solution produced by those optimization methods. These optimization methods are discussed in the next chapter.

Regarding the Agent-Based modeling, though the model is still simple, it can be easily extended to be more complex in order to fit the evacuation in reality (more detail in chapter 6). Moreover, we make a calibration so that the time of evacuation in simulation and that of real evacuation are approximately the same, which allows us to assume that: **If one solution (in this case, the sign placement) is much better than another in simulation, we believe that the better solution is also better in the real world.**

2.6.2 Open issues

2.6.2.1 How to find the best sign placement

The Agent-Based model of pedestrian is only the first step. What we need is the best sign placement which maximizes survival rate. The problem about how to find the best sign placement is discussed in the next chapter

2.6.3 Over-simlified model

The model of pedestrian evacuation is very simple, which is far from the reality where human decision is a very complex process. In order to make this simulation realistic, the agents must be enhanced more complex behaviors or at least one crowd-based behavior (the behavior which makes people form groups in evacuation)

2.6.4 Time-consuming

Although the model is very simple, it is still too slow to simulate the evacuation of the real population. The problem is how to accelerate this model and how to deal with the more complex model.

Chapter 3

Optimization of sign placement using Agent-Based Simulation

3.1 Introduction of this chapter

In the previous chapter, we have proposed a simple Agent-Based Model of pedestrian evacuation in order to evaluate how good a sign placement is. We are aware of the fact that no matter how we optimize the sign placement, we do need a model to verify the optimal placement, which means that the Agent-Based Model is always needed.

In this chapter, we present how to optimize sign placement by using the proposed model of evacuation. But first, we search the related works which make the optimization concerning evacuation from tsunami disaster.

This chapter is organize as follow: the second section is the survey on optimization of sign placement; in the third section, we present how we choose optimization method to solve our problem; in the forth section and fifth section, we present the implementation and evaluation.

3.2 Survey on optimization of sign placement

3.2.1 Minimize Average Evacuation Time

Minimize Average Evacuation Time (or MAET) (in NGUYEN et al. [2011]) is the closest problem to ours. Instead of maximizing the survival rate, the authors tried to minimize

	Location A	**Location B**	**Location C**	**AET**
Number of agents	30	30	40	
ET without sign	1800	1000	1200	1320
ET with SP1	**1000**	1000	1200	1080
ET with SP2	1800	1000	**900**	1200

TABLE 3.1: Example of Minimized Average Evacuation Time method

the evacuation time. The optimization of sign placement is modeled as an linear programming problem in finding the sign placement such that the average evacuation time of all the evacuees is minimum.

In order to clarify this approach, we present here an example in table 3.1. We have 3 locations where we want to place signs. We suppose that the populations of these locations are respectively 30%, 30%, 40%. In case of no sign, the time needed for an agent to move from location A, location B, location C to shelter are respectively 1800 seconds, 1000 seconds, 1200 seconds. In this case, the average evacuation time (AET) is 1320 seconds. If we choose to place signs at location A (ignoring other locations), the new evacuation time for each people at that location becomes **1000**. The AET of sign placement 1 is 1080 seconds. If we choose to place signs at location C (ignoring other locations), the new evacuation time for every person at location C becomes **900**. The AET of sign placement 2 is 1200 seconds. According to the objective function, this approach chooses sign placement 1 as the optimal solution because it results the Minimum Average Evacuation Time.

While this approach provides an automatic optimization way to solve our problem, it still has 2 issues to be discussed:

1. Agents' behaviors are simple and not easy to extend. In other words, it is hard to add crowd behaviors into the model in the example 3.1.

2. Average Evacuation Time and Survival Rate are different: Minimizing Average Evacuation Time seems not always to lead to the maximizing the number of survivors, which leads the optimization phase to the incorrect result by ignoring the most important factor: the tsunami arrival time. For the example in table 3.1, if the tsunami comes within 900 seconds, the sign placement 1 will save no people because the they do not succeed in reaching shelters before tsunami arrival. The optimal sign placement in this case should be the sign placement 2 which at least saves 40% of people.

We once again emphasize that the human life is the most important, and Survival Rate should be the indicator for the optimization. While the MAET provides a fast

optimization, its accuracy is still under discussion, which makes us look for another approach which would use our simulation to estimate the survival rate.

3.2.2 Other optimization concerning evacuation from tsunami

Instead of place guiding sign to lead people in evacuation, other studies have indicated that optimizing the evacuation map is also a good solution for mitigation of casualties.

Firstly, the focus of evacuation in case of tsunami is the city map. Most of studies analyze the map in order to evaluate the performance of evacuation (e.g. the studies in PÉROCHE et al. [2014], LI et al. [2014], CHURCH and COVA [2000], LIN et al. [2008]). In PÉROCHE et al. [2014], author proposed to use the Dijkstras algorithm in order to highlight the shortest path between areas at risk and to indicate evacuation areas. In LI et al. [2014], authors tried to find the optimal routes between dangerous zones and shelters by minimizing total evacuation time (which is very close to MAET proposed in NGUYEN et al. [2011]). While these works focus on tsunami evacuation, their optimization problems are far from ours.

3.2.3 Optimization by exploring parameter of simulation

According to our survey on optimization of sign placement in case of tsunami evacuation, all the studies do not fit our work. We then begin searching for the general approaches which are able to fit plenty of problems, hoping to find a solution to our problem. While in WEISE [2009], authors presented most methods for general optimization problems, authors in GOSAVI [2014] summarized the optimization methods concerning simulation.

In fact, optimization in which Agent-Based Simulation plays a role of fitness function has been widely used DUBOZ et al. [2010]. In STONEDAHL and WILENSKY [2010], the authors presented comparison among parameter-space exploiting methods (uniform random search, hill climbing and genetic algorithm). And among these methods, they argued to choose the genetic algorithm which inspired us to adapt it into our proposed formulation. Another application of genetic algorithm is introduced in PARK et al. [2012]. The authors proposed to build vertical shelters (the high building) for evacuees to stay from tsunami. Their problem is to find locations to build limited high buildings in order to have as many survivors as possible, which is similar to our problem. The most interesting idea of this work is the approach to do optimization. They build the simulation and find the optimal locations by exploring the parameters which are the locations of buildings in this case. Since their problem is similar to ours, we decided to study their approach. Precisely, the authors presented their work as a problem of

optimizing shelter placements. Their goal is to find out the locations of a limited number of shelters (called $N_{Shelter}$) in order to maximize the number of survivors. The solution is represented by a list of 2-dimension coordinates of shelters. These solutions plays a role of chromosomes in the Genetic Algorithm that they proposed to do the optimization job. The fitness is the number of survivors which is computed by a formulation between evacuation speed and initial locations of each evacuee.

3.3 Chosen optimization method

We represent our problem in a mathematical way in order to clarify our arguments.

$$max_{\theta \in \Theta} SR(\theta)$$

$$SR(\theta) = \sum RunSimulation(\Theta)/N_{RTS}$$

Our problem is to maximize $SR(\theta)$, in which parameter θ is the sign placement (SP) and $SR(\theta)$ is the survival rate. The searching space is $\Theta = \nVdash..\ltimes^K$, where K is the number of signs, and where n is the number of intersections, because each sign is associated with a location (n possibilities) and a direction (also n possibilities). We do not have exact survival rate. Then, we run the simulation (N_{RTS} - "**R**epeating **T**imes of **S**imulation") times for one sign placement. The survival rate of this sign placement is the average of all of outcomes from simulation. In our work, we choose a genetic algorithm (a meta-heuristics algorithm) to do optimization for 2 reasons:

Our first reason comes from WEISE [2009]. Our objective function is the survival rate resulted from an Agent-Based simulation which is complex and stochastic. The objective function is like a black box which is independent from the optimization phase. According to authors in WEISE [2009], the solution for black box objective function is the meta-heuristic algorithm. The figure 3.1 shows the nature of the optimization phase in which the objective function is a black box. In fact, there is no way to know in advance which is the better solution for the next step of searching. The only indicator here is the survival rate which plays the roles as heuristic function. Thus, the meta-heuristic algorithm is the most appropriate approach for the optimization in which the objective function comes from Agent-Based simulation.

The second reason for us to choose meta-heuristic algorithm is consulted by the study in BARTON [2009]. The authors of this study summarized all strategies to solve this kind of problem. Each strategy depends on which kind of the parameters are and how large the parameter space is. The figure 3.2 presents our strategy to choose optimization

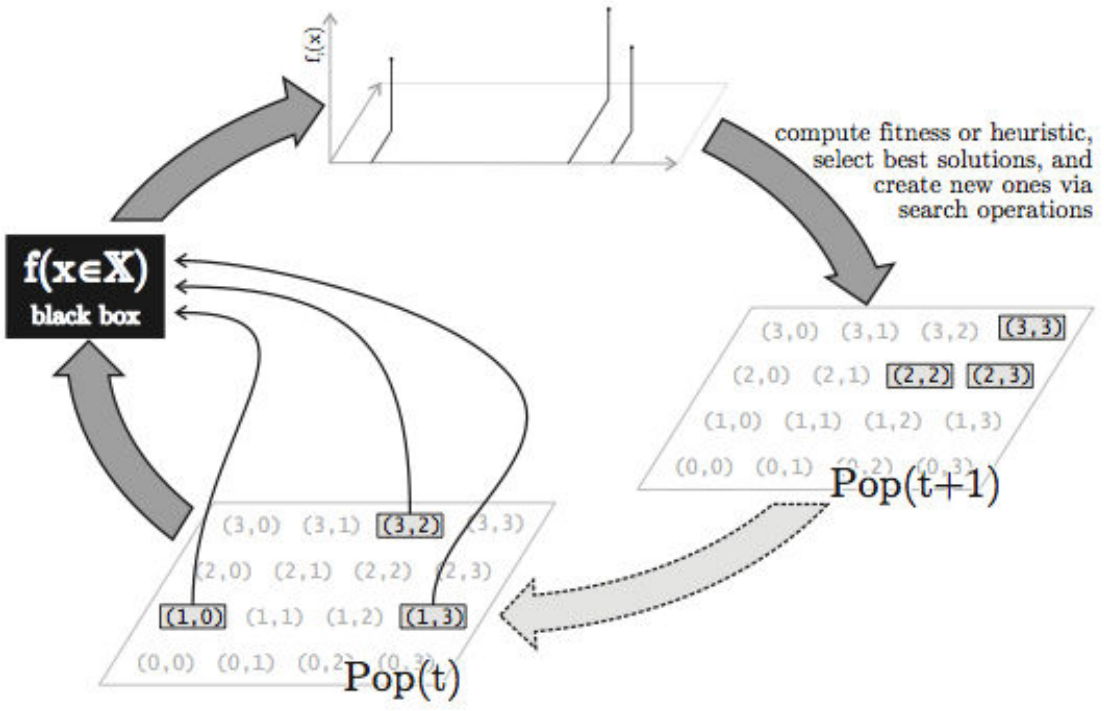


FIGURE 3.1: Optimization with fitness computed by black box operation (figure from WEISE [2009])

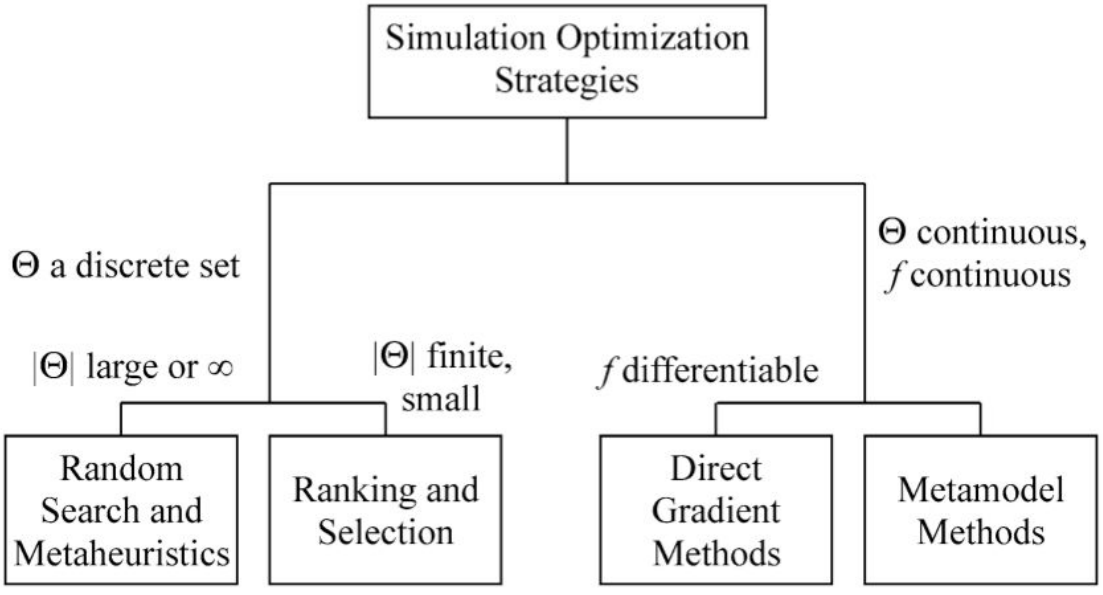


FIGURE 3.2: Simulation optimization strategy depends on the nature of Θ and f (figure from BARTON and MECKESHEIMER [2006])

method. Since the parameters of our model is discrete and the parameter space is very large, we proposed to use Metaheuristics Algorithm.

Thus, we use genetic algorithm to solve the problem of optimizing sign placements in case of tsunami evacuation.

Procedure Genetic Algorithm
begin (1)
  $t = 0$;
  initialize $P(t)$;
  evaluate $P(t)$;
  While (Not *termination-condition*) **do**
  begin (2)
    $t = t + 1$;
    select $P(t)$ *from* $P(t-1)$;
    recombine $P(t)$;
    evaluate $P(t)$;
  end (2)
end (1)

FIGURE 3.3: Typical structure of Genetic Algorithm (figure from HERRERA et al. [1998])

3.4 Implementation

We use the typical genetic algorithm (described in HERRERA et al. [1998]). Figure 3.3 presents the genetic algorithm structure, which includes 4 parts: initialization, selection, evaluation, recombination (or breeding). We already have the evaluation by Agent-Based simulation. The survival rate resulting from the simulation becomes fitness of the chromosome. Other 3 parts are defined as follow:

3.4.1 The search space

As we mentioned in the previous chapter, the map of the city is modeled as a graph G=(V, A) in which V is the set of vertices and A is the set of arcs. The figure 3.4 presents an example of a block of the city, which includes:

- 7 vertices ($|V| = 7$)
- 2 shelter vertices
- 17 arcs ($|A| = 17$)

In order to represent the chromosome, we consult the research in FRANZ [2006]. A chromosome in our case represents a solution which is exactly a sign placement of K signs. Knowing that the coefficient K is the most important parameter for this problem, which means that every encoded chromosome must not contain more than K signs. We propose to use a set to define a chromosome which contains K different integers. Each value represents the index of the arc representing the road (including the direction) where the sign is placed. For the example in figure 3.4, the sign placement is composed

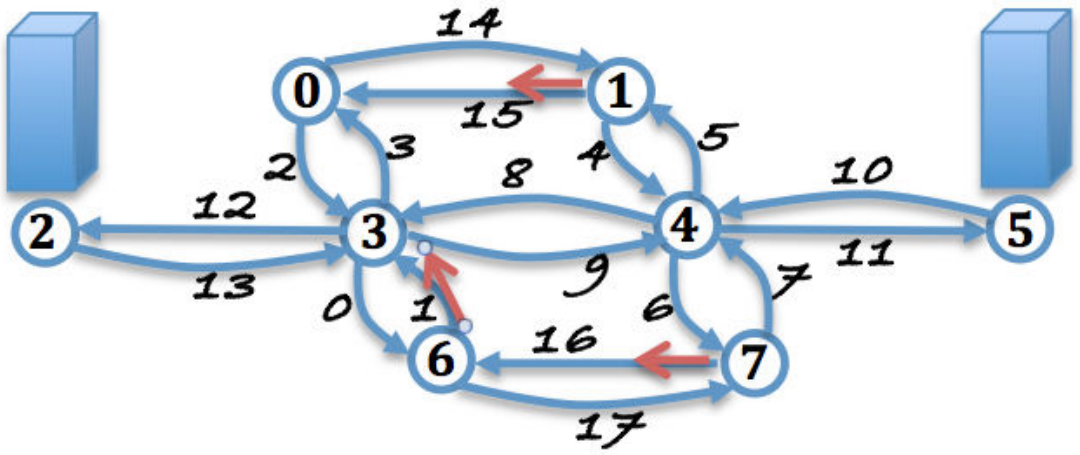


FIGURE 3.4: An example graph of a block in the city and a sign placement of 3 signs

of 3 signs (K = 3) whose arc indices are respectively 1, 15, 16. Then the chromosome representing this sign placement is defined a set SP ={1,15,16}.

3.4.2 Selection

We decide to use "Fitness proportionate selection", also known as "roulette wheel selection", described in BACK [1996]. In order to clarify the selection strategy, we introduce an example that: we assume to have a population of 4 individuals ($\mathcal{P} = 4$) whose fitnesses are respectively $f_1 = 0.12$, $f_2 = 0.24$, $f_3 = 0.38$, $f_4 = 0.46$ (corresponding survival rates calculated by simulation). The probabilities for individual i to be selected is calculated by the formula $p_i = \frac{f_i}{\sum_{j=1}^{\mathcal{P}} f_j}$. Then, the probabilities of these individuals are respectively 0.1, 0.2, 0.32, 0.38.

3.4.3 Recombination

In order to simplify the optimization, we begin with the simple recombination with default crossover (1-point crossover) and default mutation (Uniform mutation). The crossover operator in this case is one-point crossover which separates the parents chromosome into two parts and combines them to create two new offsprings. The chosen mutation operator is uniform mutation which uniformly chooses another sign (other index) to replace and random sign in the chromosome.

Parameter name	**Symbol**	**Value**	**Unit**
Number of agents	M	1000	agents
Number of vertices	$\|V\|$	65	vertices
Number of arcs	$\|A\|$	184	arcs
Number of signs	K	10	signs
Maximum speed of pedestrian	$\mathcal{V}_{max}$	1.5	m/s
Number of **R**epeating **T**imes of **S**imulation	N_{RTS}	10	times

TABLE 3.2: Fixed parameters of the Agent-Based model of pedestrian evacuation

3.4.4 Initialization and Termination condition

Termination condition is the important factor for the genetic algorithm. Being aware that the genetic algorithm needs very much time to reach the convergence, we choose different conditions for termination depending on a specific experiment.

3.5 Experimentation

3.5.1 Description of model

As we mention in the previous section, we use Agent-Based simulation to evaluate the sign placement. The model (which is discussed in Chapter 2) is totally reused except the size of the map. As we state in the experimentation section of the chapter 2, the Agent-Based simulation is time consuming. The application of genetic algorithm to the whole map of the city becomes infeasible. In fact, it takes the Agent-Based simulation more than 10 minutes to evaluate one sign placement. In this case, we have to run the Genetic Algorithm on 100 generations and with 100 individuals per generations. Then, it would take nearly 3 months to finish the optimization. That is why we run the evacuation only on a ward instead of the whole city. The figure 3.5 shows the map of the small ward where we implement evacuation. The map contains only 65 crossroads (corresponding 65 vertices in the modeled graph).

In this case, we simulate the evacuation of the worst scenario which determines the evacuation time to be 15 minutes. In fact, according to the study in VU and NGUYEN [2008], this scenario assumes that even when the earthquake center is far from the coast, the local government needs time to decide if there is a tsunami, then to declare the alert. The spreading information also takes time to reach all the people. Meanwhile, the tsunami continues spreading. Then, the evacuation is triggered when the first wave of tsunami is very near the coast. The evacuation time for people shortens to 15 minutes. Other parameters of the model are fixed as the description in the table 3.2.

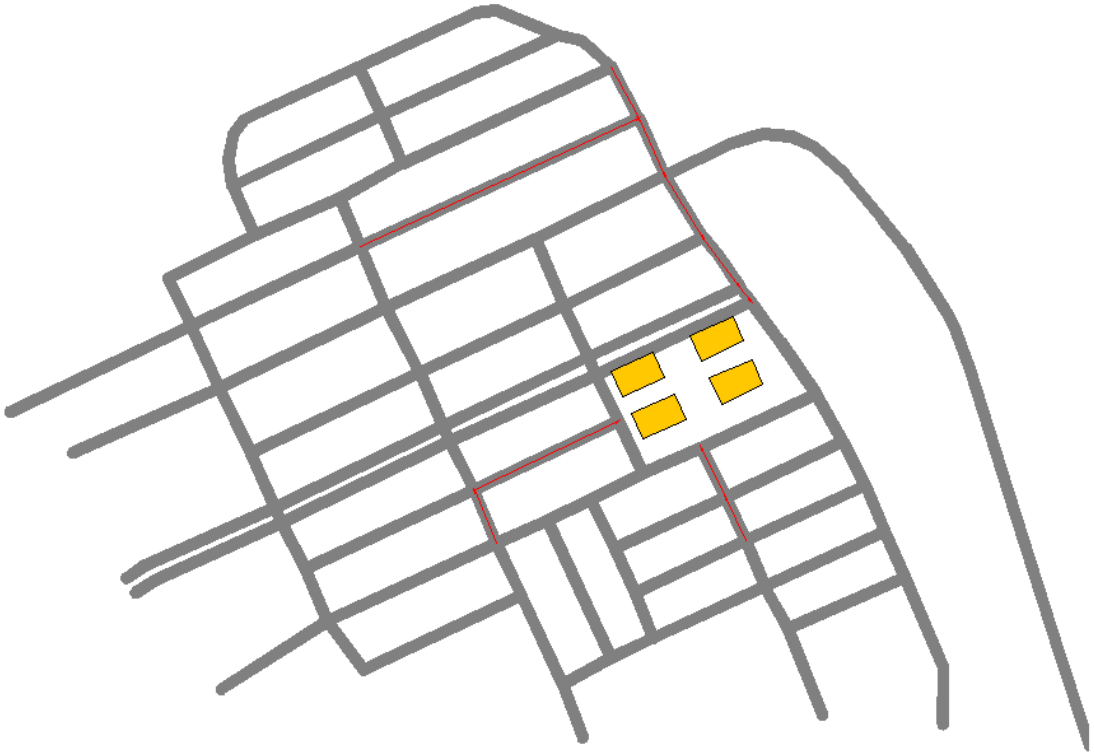


FIGURE 3.5: Map of a ward where the evacuation takes place

Parameter name	Symbol	Value	Unit
Number of individuals	$\mathcal{P}$	100	individuals
Probability of crossover		0.7	
Probability of mutation		0.1	

TABLE 3.3: Fixed parameters of Genetic Algorithm

3.5.2 Description of genetic algorithm

In this case, we considerably reduce the size of the evacuation environment as well as the number of agents. We run the genetic algorithm during 12 hours. The recombination method is the same as mentioned in the previous section. Other coefficients are predefined in the table 3.3.

3.5.3 Performance of Genetic Algorithm

The purpose of this experiment is how many people are saved in the evacuation by the optimal sign placement computed by Genetic Algorithm. The figure 3.6 shows the result of the experiment. According to the result, the optimization phase is also time-consuming as the Agent-Based simulation is, even though the environment is reduced significantly.

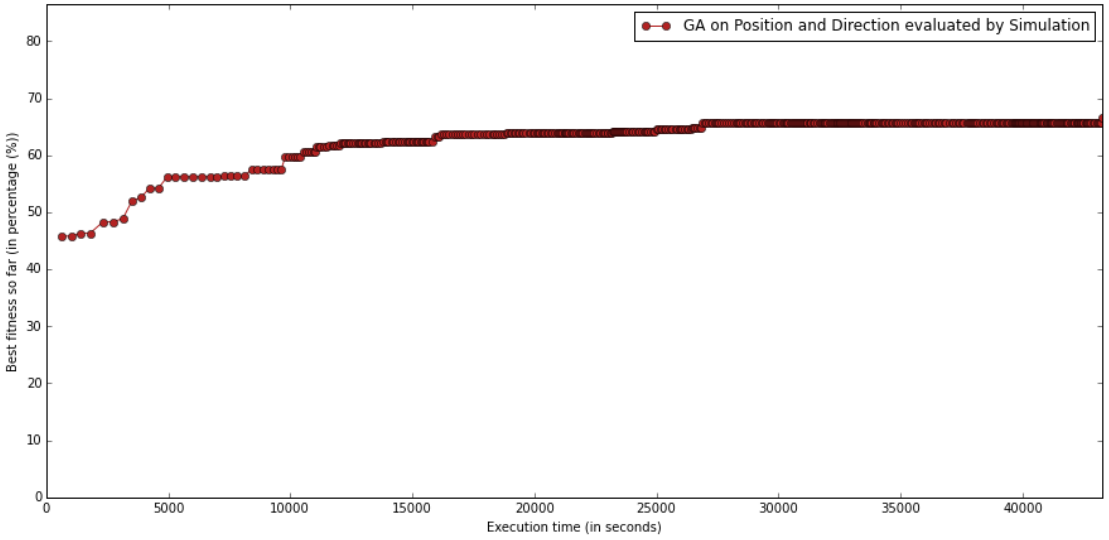


FIGURE 3.6: Result of optimization of sign placement with Genetic Algorithm. Method: chromosome coded by position and location, fitness evaluated by simulation

This makes the proposed optimization method infeasible in case of evacuation of the full map with the real population.

3.5.4 Comparison

As we mention in the previous chapter, no matter how we do optimization, we do need a Agent-Based simulation to re-verify the optimal result. In this case, we run two optimization algorithms corresponding two approaches with different values of limited evacuation time:

- Minimized Average Evacuation Time
- Genetic Algorithm with chromosome encoded by sign position and sign direction, fitness calculated by Agent-Based simulation

The results from both optimization methods are two sign placements. Each sign placement contains 10 signs. We run again the simulation over two sign placements. Then the survival rates from the simulations are compared.

As we mentioned in the previous section, we focus on the scenario where the evacuation time is very short (less than 15 minutes). Then, the solutions for limited evacuation time less than 15 minutes (or less than 900 minutes) are important. The figure 3.7 shows the result of two approaches. When the evacuation time is more than 900 seconds, the results of two approaches are approximately equal. But when the limited evacuation time is less than 900 seconds, the survival rates from MAET drop considerably. When

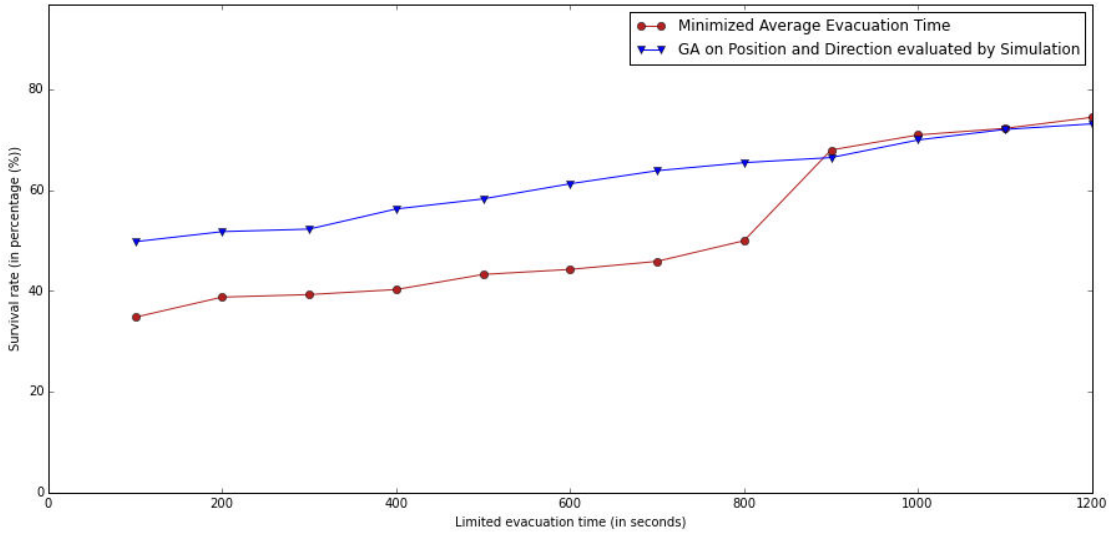


FIGURE 3.7: Genetic Algorithm. Method: chromosome coded by position and location, fitness evaluated by simulation VS Minimizing Average Evacuation Time

the limited evacuation time is less than 900 seconds, we can see in the figure 3.7 that the Genetic Algorithm prevails the MAET. An example is the evacuation time is 800 seconds, the best solution of MAET can safe only 47% people but the best solution of Genetic Algorithm save more than 60%. The only reason for this result is that the MAET does not take into account the limited evacuation time, but the Genetic Algorithm does. The optimization only focuses on the average evacuation of all the agents, which still allocates signs for the area where people cannot reach shelters before tsunami arrival. This experimentation once again confirms that the approach MAET does not give the reliable results.

3.6 Conclusion

3.6.1 Proposition of optimization method

In this chapter, we have proposed a method to optimize sign placement in order to guide people in evacuation from tsunami disaster. This method uses genetic algorithm (a meta-heuristic algorithm) to explore parameter of evacuation simulation. It is true that all the approaches used in our solution are already proposed by other scholars. The only contribution is the combination of these existed approaches to apply to the problem of optimizing sign placements for tsunami evacuation.

3.6.2 Ineffective optimization

The experimentation results show that this optimization method is still ineffective even when the model of evacuation is very simple. Two reasons for this inefficiency are:

1. Time-consuming simulation: The Agent-Based simulation takes much time to evaluate the fitness of chromosome. In this case, the chromosome represents the a sign placement, the fitness is the survival rate of evacuation simulation.

2. The way representing chromosome is naive which prevents the optimization from reaching the convergence. The chromosome here is coded by using the arc which would make the signs form the cycle. When an adding sign forming a cycle, the survival rate declines badly. This is in sharp contrast to what we expect that the more signs are added, the more people survive.

In order to bring this kind of optimization into application, we need an efficient method to optimize the sign placement.

3.6.3 Subproblems to solve

At the end of this chapter, our proposed method opens two subproblems to solve in the next chapters:

1. Complex model of evacuation: As we proposed in chapter 2, the model of pedestrian evacuation is very simple. The evacuees' behaviors in the reality is much more complex, which requires a complex model of pedestrians. In fact, the most important actions of evacuees tend to form groups, which means that a reliable model of evacuation needs at least one crowd-based behavior.

2. Tractable optimization: Though the evaluation uses the simple model, the optimization phase is still too slow to reach the optimal solution. All we need is Acceleration of evaluation of sign placement.

Chapter 4

A Linear Programming Approach to compute Survival Rate for Simple Agents

4.1 Introduction

In the previous chapter, we have proposed a method to optimize sign placement, this method uses an meta-heuristic algorithm whose heuristic function is computed by Agent-Based simulation. Concretely, the optimizing method used genetic algorithm in which the chromosomes represent the arcs where the signs are places and the fitness is the survival rate. However this proposed method is still far from application due to: time-consuming simulation and naive representation of chromosomes.

The current chapter is one of two most important chapters of this document. In this chapter, we propose a method to optimize an approximation of the survival rate, which does not rely on a simulation. Thus this approach drastically speeds up the optimization process. Before going to the details of the method, we propose another different aspect of human behaviors in evacuation and another approach to model the pedestrian evacuation rather than the Agent-Based approach. Then, we propose a liner programming model which evaluate very fast the survival rate of a sign placement.

This chapter is organize as follow: the second section is the survey on Surrogate model; in the third section, we present our proposed surrogate model (linear programming model based on Markov chain decision); in the forth section and the fifth section, we present the implementation and evaluation.

4.1.1 Time consuming Agent-Based simulation

According to the experimentation, we find that Agent-Based Simulation takes much time to evaluate survival rate. In order to clarify how long the Agent-Based Simulation would take to find out the best sign placement, we present an example.

In this example, we simulate 30 minutes (1800 seconds) of an agent-based evacuation of one million (10^6) pedestrians in a big city such as Danang City in Vietnam): with regard to agent-based approach, we must simulate behaviors of every single pedestrian; we suppose that at every second of simulation, each agent makes its decision on where to go; this simulation then has to run 18×10^8 decisions; since the simulation results are stochastic, we run the same simulation with the same sign placement 100 times and take the average survival rate as result. As a result, only one sign placement needs 18×10^{10} decisions. In case of parameter exploring by genetic algorithm, this simulation should normally be repeated 100 times (for the different parameters corresponding 100 chromosomes) to evaluate fitness and be repeated through 100 generations; then the system must pass 18×10^{14} decisions; for a single personal computer, if we suppose that it can simulate one million pedestrian decisions within one second, then it takes 5×10^5 hours. Thus, the agent-based simulation seems not to be realistic in this case, we really need an approach to speed up the evaluation. The new approach must reproduce almost the same result as the agent-base one does in the same scenario and also the same input parameters.

4.2 Survey on surrogate model

The direct idea to overcome the time-consuming Agent-Based model is to replace it by another faster model. The new model (replacing the old one) is usually called surrogate. Following this idea, we make a short survey on how others made the surrogate in order to accelerate their models.

We have found the general idea of abstraction in SAITTA and ZUCKER [2013], BARTON [2009], which presented the same purpose as what we wanted. In the figure 4.1, the authors (in SAITTA and ZUCKER [2013]) presented the Abstract process for Problem solving. Concretely, instead of directly solving the problem with costly computational effort, we could indirectly solve it through an Abstract process including 3 steps:

- Representing the ground problem into another abstract version in order to reduce computational cost

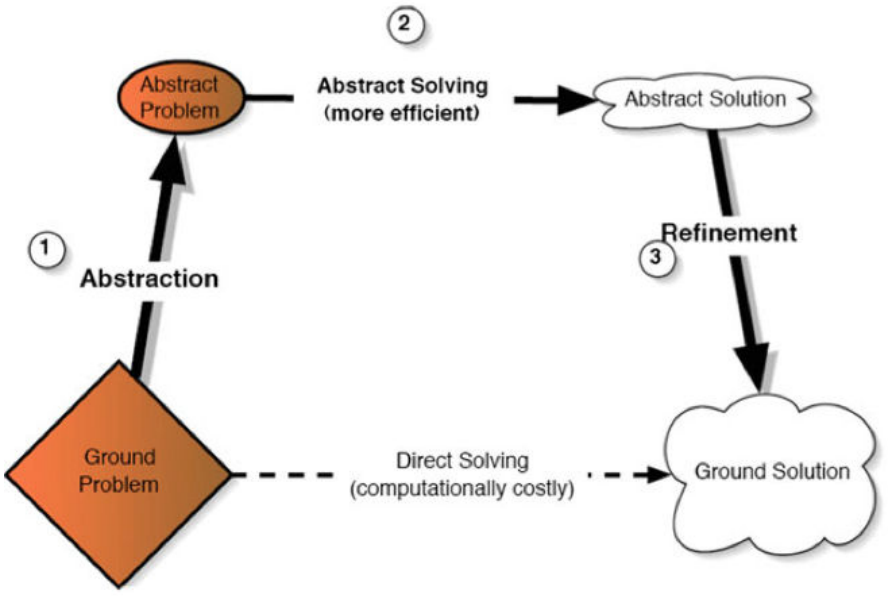


FIGURE 4.1: General idea of abstraction in SAITTA and ZUCKER [2013] page 7

- Solving the abstract problem with more efficiency
- Refining the solution to get the ground solution

While the idea of surrogate is simple, it is widely used in plenty of optimization problems and in various domains. In COZAD et al. [2014], authors proposed a surrogate model in their "Carbon Capture Adsorber" Case Study in order to accelerate their "synthesis of an optimal carbon capture process". For an Agent-Based modeling, authors in SHIRAZI et al. [2014] have proposed an approach which clusters a group of agents into a single agent (called meta-agent), which reduced the number of agents. The application of this approach was the simulation of the movement of the blood cells in the veins.

4.3 A Linear Programming Approach to compute Survival Rate for Simple Agents

In this section, we propose a surrogate model replacing Agent-Based Model of pedestrian evacuation. The surrogate model is in fact a linear program in which the evacuation is modeled as a Markov chain.

4.3.1 Modeling agent decision as Markov decision process

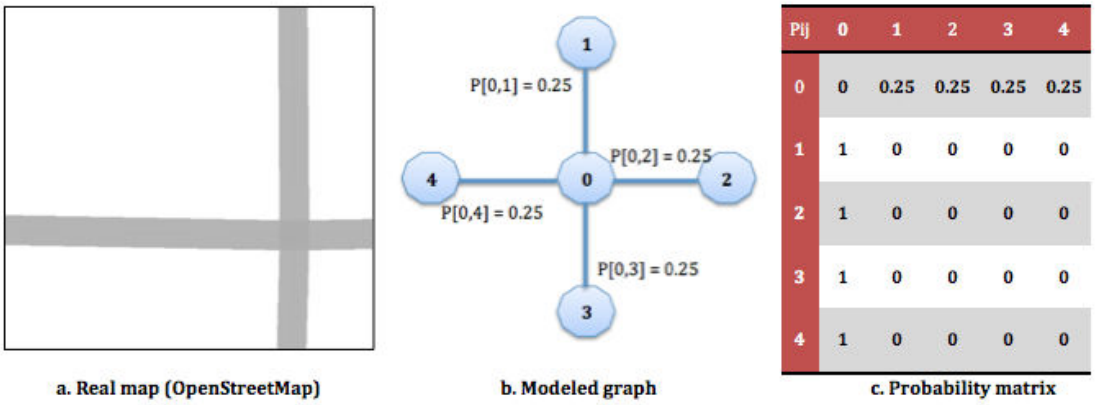
As we described in the simplest model, the agents' behaviors are just reactive. If there is a sign at the crossroad, agents take turn toward the direction of the sign. Otherwise, agents take turn randomly. This simple decision leads us to think about the probability of the direction toward which agents should turn. Specifically, if the there is a sign, the probability for agents to turn toward the direction of the sign is equal to 1.0 and the probabilities of other directions are 0.0. If there is no sign, the probabilities of every direction are all equal 1/(Number of directions). The way of representation of decision into probabilities motivates us to use Markov Chain. The following example clarifies the idea of this representation.

The figure 4.2 and the figure 4.3 both describe an example of how a agent decision (in Agent-Based model) is modeled as a Markov decision. The crossroad in the example is modeled as a graph of 5 vertices and 4 edges (corresponding 8 arcs). While the figure 4.2 represents the case that no sign is placed, the figure 4.3 shows the case that one sign is placed at the crossroad and that the direction of the sign heads toward the right.

In the case that no sign is placed (in figure 4.2), if an agent (representing a pedestrian evacuee) situates at the vertex 0, it takes a turn randomly. Then, the probabilities for that agent to turn to vertex 1, vertex 2, vertex 3, vertex 4 are all equal to 0.25 (equal to 1/4). The probability transition matrix is then predefined in figure 4.2c in which the values of the cells in row 0 are P[0,1]=P[0,2]=P[0,3]=P[0,4]=0.25=1/4. If an agent is at the vertex 1 or vertex 2 or vertex 3 or vertex 4, its only choice is the vertex 0, which means that the probabilities for that agent to turn to vertex 0 in these cases are all equal to 100%. And in the probability transition matrix (in figure 4.2c), the values of cells column 0 are P[1,0]=P[2,0]=P[3,0]=P[4,0]=1=100%. The values of other cells are all equal to 0 because there are no direct roads connecting any pair of these vertices.

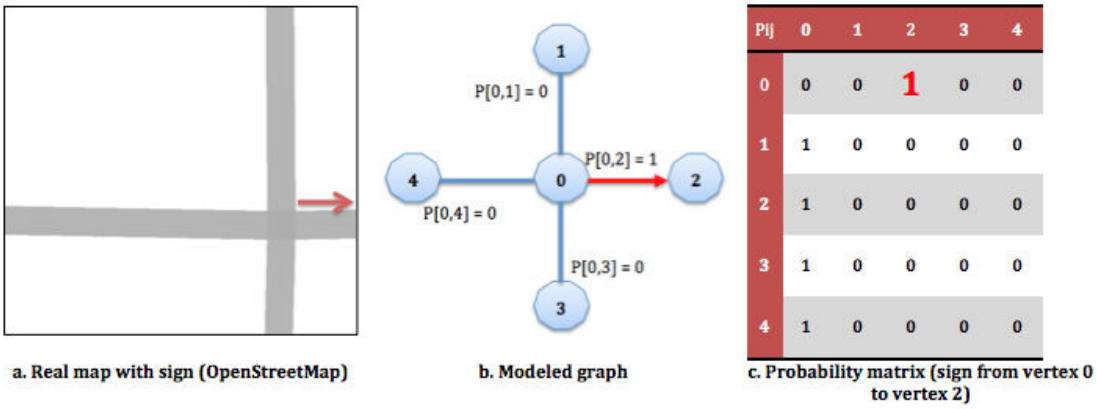
In the figure 4.3, a sign is placed. The position of the sign is vertex 0 and its direction points toward vertex 2. In the real world, any evacuees situating at vertex 0 follow the sign to turn to vertex 2. In the simulation, 100% of agents at vertex 0 turn to vertex 2, which means that the probability of agents at vertex 0 to turn to vertex 2 is 100%. This modifies the values of the probability transition matrix (figure 4.3c) in which the value P[0,2] = 1; and the P[0,0]=P[2,0]=P[3,0]=P[4,0].

It is true for the evacuating movement that a pedestrian makes a sequence of decisions to turn from a crossroad to another. From the aspect of probability for an agent to turn from a vertex to another, the sequence of decisions is considered as a Markov chain. Before modeling the pedestrians evacuation into Markov Chain model, we present a short description of Markov Chain modeling.



Pij	0	1	2	3	4
0	0	0.25	0.25	0.25	0.25
1	1	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	1	0	0	0	0

FIGURE 4.2: Example of Markov based decision in case with out sign



Pij	0	1	2	3	4
0	0	0	1	0	0
1	1	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	1	0	0	0	0

FIGURE 4.3: Example of Markov based decision in case with 1 sign from vertex 0 to vertex 2

Let a graph G = (V, A) represents the map in which V = 1, 2,...,n is a set of junctions and A $\subseteq$ V $\times$ V is a set of arcs representing directed roads connecting vertices. Each arc is associated with a weight c_{ij} representing the required time for an agent to move from vertex i to vertex j. Also, the set of neighbors of vertex i is referred to as N(i) = {j: (i, j) $\in$ A}. Let X $\subset$ V denote the set of shelters (which represent the high buildings or the high-ground places to which the people evacuate in case of tsunami). If an agent reaches a shelter, then it is considered as out of danger.

A Markov chain is composed of an initial distribution $\mu = \{\mu_i$: i $\in$ V$\}$ and a stochastic n$\times$n matrix P representing the transition probabilities from one vertex to another. At time t = 0, each agent is randomly placed on the graph with respect to the given distribution μ. At time $t > 0$, an agent positioned on vertex i will move to vertex j $\in$ N(i) with the probability p_{ij} such that if (i, j) $\notin$ A then $p_{ij} = 0$ and such that $\sum_{j \in V} p_{ij} = 1$.

Eventually, these probabilities will be estimated by using more sophisticated crowd models. Here, we simply use uniform probabilities. If there is a sign from vertex i

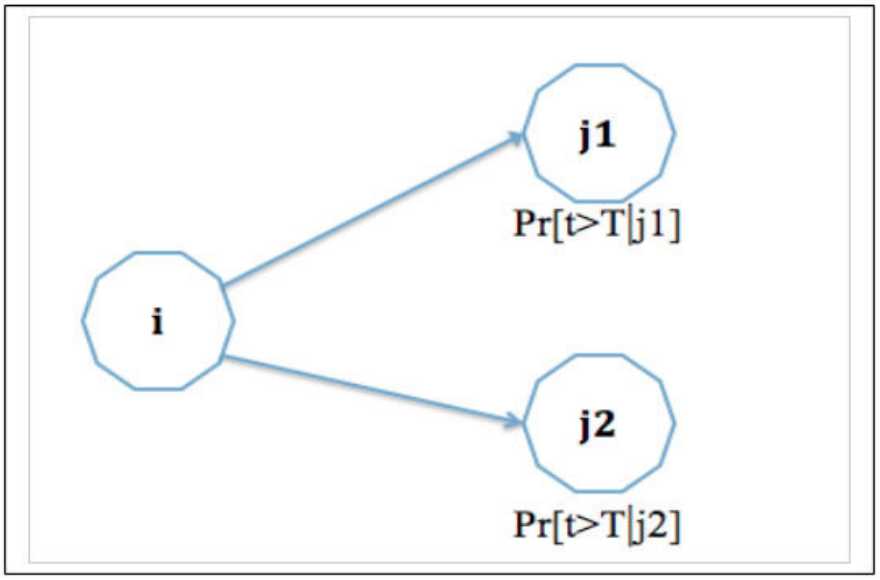


FIGURE 4.4: Example of distribution probability for 3 vertices

pointing to j then $p_{ij} = 1$ and $p_{ik} = 0$, $\forall$k $\neq$ j. Thus, a quadruplet (G, X, μ, P) will completely define the territory and the behavior of agents. We assume that every agent has a threshold of time to evacuate, the optimizing problem becomes to find k arcs (from A) to place sign such that the number of agents whose evacuation time is above some threshold (e.g. 30 minutes) is maximized.

4.3.2 Linear Programming Formulation of Casualties Evaluation of Pedestrian Evacuation

With the Markov Chain modeling, the crowd simulation is stochastic, so we cannot know in advance the exact time required for some agents to reach a shelter. Instead, we can compute the distribution probability over its evacuation time. More precisely, we will compute the probability for an agent starting on vertex i to reach a shelter in more than T seconds. Let us call this probability $Pr[t > T|i]$.

Let us make it clear with an example (Figure 4.4)

Assuming for the sake of simplicity that the time required for an agent to move from i to j1(or j2) is 1 second and that $p_{i,j1} = p_{i,j2}$=1/2. Also assuming that we know $Pr[t > T|j1]$ and $Pr[t > T|j2]$. Then, we can compute the same probability for vertex i. To reach a shelter from vertex i, we need 1 more second than from vertex j1 or j2. So if $T > 1$, we have

$$Pr[t > T|i] = \frac{1}{2}Pr[t > T - 1|j1] + \frac{1}{2}Pr[t > T - 1|j2]$$

Finally, assuming μ_i refers to the number of agents on vertex i at time 0, the expected number of casualties will be:

$$\sum_i \mu_i Pr[t > T|i]$$

The probability distribution $P[t > T|i]$ is a continuous function of T. We need to discretize this function to compute its value. Let α be the discretization step and K be an integer such that Kα is the evacuation deadline (the maximum time for a pedestrian to reach a shelter). The weigh of each arc (i, j) is then predefined as $\hat{c}_{i,j} = \lceil \frac{c_{i,j}}{\alpha} \rceil$. For all i $\in$ V, define the variable $q_{i,k}$ as the probability for an agent starting at vertex i to reach the shelter in time t $\geq$ kα. Let [x]+ be x if $x > 0$ or be 0 otherwise. The equations for $q_{i,k}$ are then defined as follows:

1. For all the pedestrians positioned at any vertices, they would sooner or later arrive at shelters then
$$\forall i \in V, q_{i,0} = 1 \ (1)$$

2. For the pedestrians who stay at the shelter, there is no need to move then
$$\forall i \in X, \forall k \geq 1, q_{i,k} = 0 \ (2)$$

3. For the pedestrians who stay at a vertex i (not the shelter), if they choose vertex j (j $\in$ N(i)) for the next target, their probability to reach shelter in time t $\geq$ kα will be the product of the transition probabilities from i to j (p_{ij}) and the probability to reach shelters in time $t \geq (k - \hat{c}_{ij})\alpha$. Then in this case, the general probability for the pedestrians to reach shelters is the scalar product of the two probabilities with the respect of all the neighbors of i. The equation then becomes
$$\forall i \in V \setminus X, \forall k \geq 0, q_{i,k} = \sum_{j \in N(i)} p_{ij} q_{j,[k-\hat{c}_{ij}]_+} \ (3)$$

Thus, with μ_i is the number of pedestrians starting on vertex i, we can compute the expected number of deaths with the following linear program:

$$\sum_{i \in V} \mu_i q_{i,K} \quad \textit{such that}:$$

$$\begin{aligned} \forall i \in V, & \quad q_{i,0} = 1 \\ \forall i \in X, \forall k \geq 1, & \quad q_{i,k} = 0 \\ \forall i \in V \setminus X, \forall k \geq 0, & \quad q_{i,k} = \sum_{j \in N(i)} p_{ij} q_{j,[k-\hat{c}_{ij}]_+} \end{aligned}$$

Finally, the number of survivors is simply the total number of agents minus the number of deaths.

4.4 Implementation

The important idea for the evaluation of a surrogate model is that the Agent-Based model plays the role as a reference. Our purpose is that the estimated survival rates in the surrogate model should be as close to those of Agent-Based model as possible. Larger the gap between the results of two models are, worse the surrogate model is considered. Then, we build 2 implementations of 2 models:

- Agent-Based Model which simulates pedestrian evacuation of tsunami: This model is described in chapter 2.
- Linear Programming Model based on Markov Chain which simulates the pedestrian decisions: The model is implemented and solved with IBM CPLEX (free academic version described in D'AMBROSIO and LODI [2011]) .

The common coefficients and parameters of both models are set the same values, which guarantees the reliability of the results. Since the Agent-Based model needs much time to finish the experiment, the size of the map and the number of agents are set with various values depending on each case of experiment.

4.5 Evaluation and comparison

4.5.1 Consistency of Linear Programming model with Agent-Based model

In this first test case, we evaluate if our proposed surrogate model and Agent-Based model produce the same results. Since the Agent-Based simulation is highly stochastic, we consider our linear programming computation as consistent if the survival rate

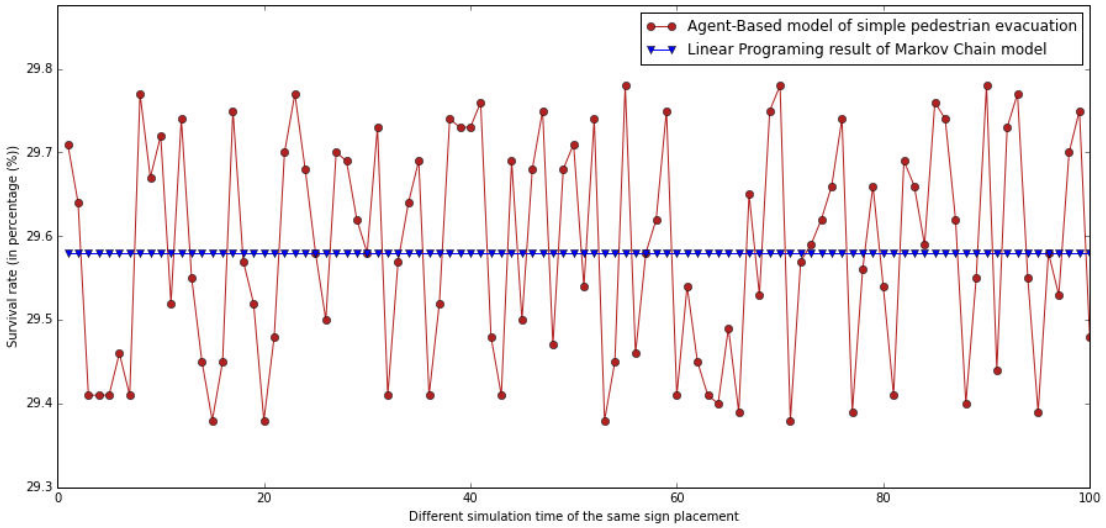


FIGURE 4.5: Consistency of Surrogate model with Agent-Based model: An example of a sign placement

computed by linear programming is equal in expectation to the average survival rate computed in 100 times of simulation, for any scenarios.

In this case, we simulate the full map of the city. We let both implementations execute 100 times with 120 different scenarios (different signs (number and locations), different thresholds of evacuation time) then we compare the returned survival rate. The value of discretization coefficient α is set as 1 (the impact of this coefficient is introduced in the next evaluation). The results of all cases of test are almost the same. While the figure 4.5 shows an example of a sign placement which is evaluated by both models, the figure 4.6 shows the consistency of survival rates (evaluated by both models) of all tested sign placement. In the figure 4.6, the points are distributed near the secondary diagonal, which means that the Surrogate model and the Agent-Based model give the same results. While the Agent-Base simulation returns the various results on each execution, the average of these results is almost the same as that given by the linear programming formulation.

The most important conclusion for this experimental result is that the we can replace the Agent-Based model of pedestrian evacuation by the surrogate model of Markov chain decision formulated by Linear Programming. The next evaluation focuses on how fast the surrogate model is.

4.5.2 Impact of discretization step on computational speed

As we mention in the previous section, the discretization step has a huge impact on both the speed and the quality of the linear programming result. We tested our linear

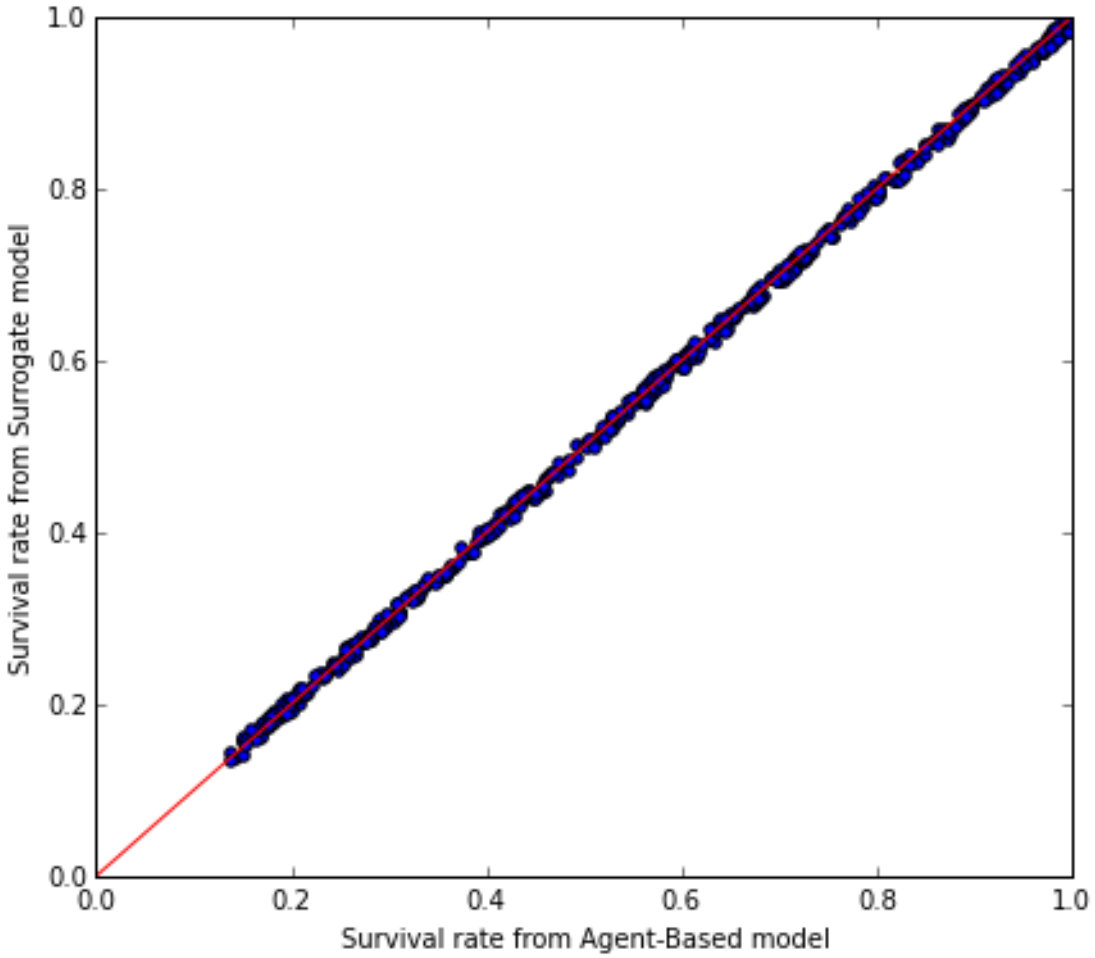


FIGURE 4.6: Consistency of Surrogate model with Agent-Based model: all tested sign placements

programming implementation with different α, and concluded that the greater α, the more execution time is significantly reduced (figure 4.7) but the more error is gradually produced on output result (figure 4.8).

In figure 4.8, let #LP be the survival rate of linear programming, and #AB is that of Agent-Based simulation; 3% of error means that #AB$\times$0.97 $<$ #LP $<$ #AB$\times$1.03; 0% means that the linear programming result is equal to that of the simulation.

The reason is that the greater value of α reduced the number of variables $q_{i,k}$. In this case, we proposed to use $\alpha = 5.0$ to minimize the execution time but to limit the error less than 1%.

4.5.3 Comparing execution time of a single run of both models

In experimental setting, we compare which model (between the surrogate model and the Agent-Based model) is faster and how fast it is. We run again the same experiment as

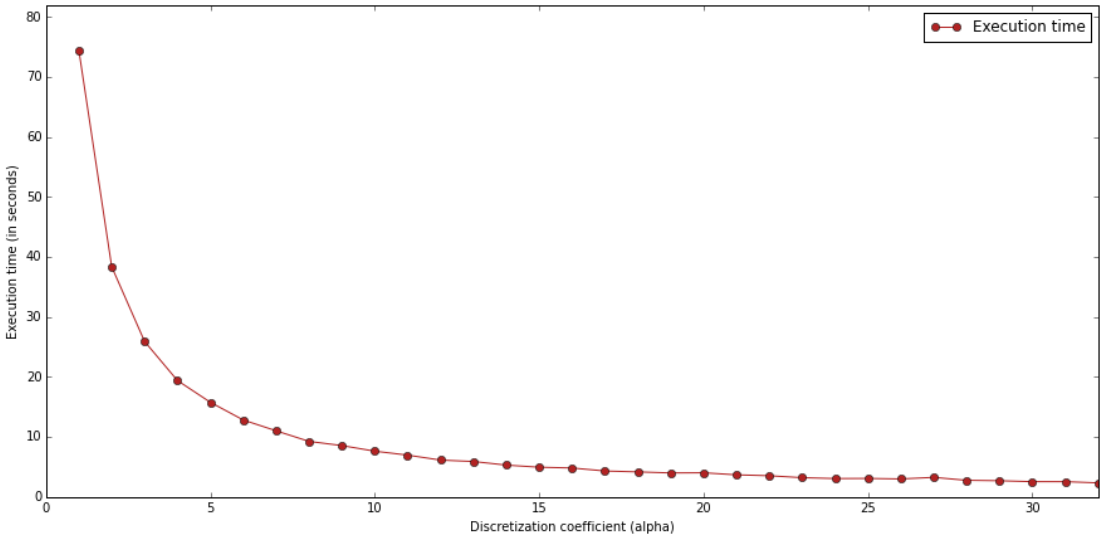


FIGURE 4.7: Execution time and discretization coefficient α

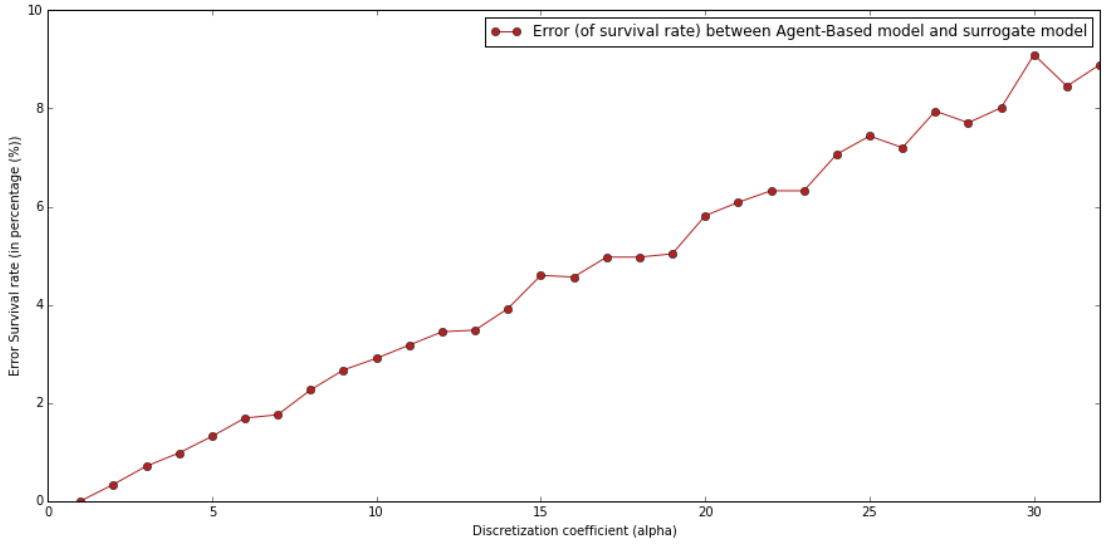


FIGURE 4.8: Impact of discretization coefficient α on error between Agent-Based model and Linear Programming model

Chapter 2, section 5.2 (called experiment 2.5.2) which shows the impact of number of agents on execution time of Agent-Based Simulation. The parameters for experiment in this case are set as those of the experiment 2.5.2 in which: the graph representing the real map of the city (figure 2.4), the evacuation time is 7200 steps corresponding to 2 hours in the real world. The experiment shows that more agents slow down the Agent-Based simulation (in figure 2.6). We run the surrogate mode (Linear Programming model based on Markov Chain which simulates the pedestrian decisions) with the same values of all common coefficients and parameters of both models, and then note the execution time.

With respect to execution time, the figure 4.9 shows the results of both models. The

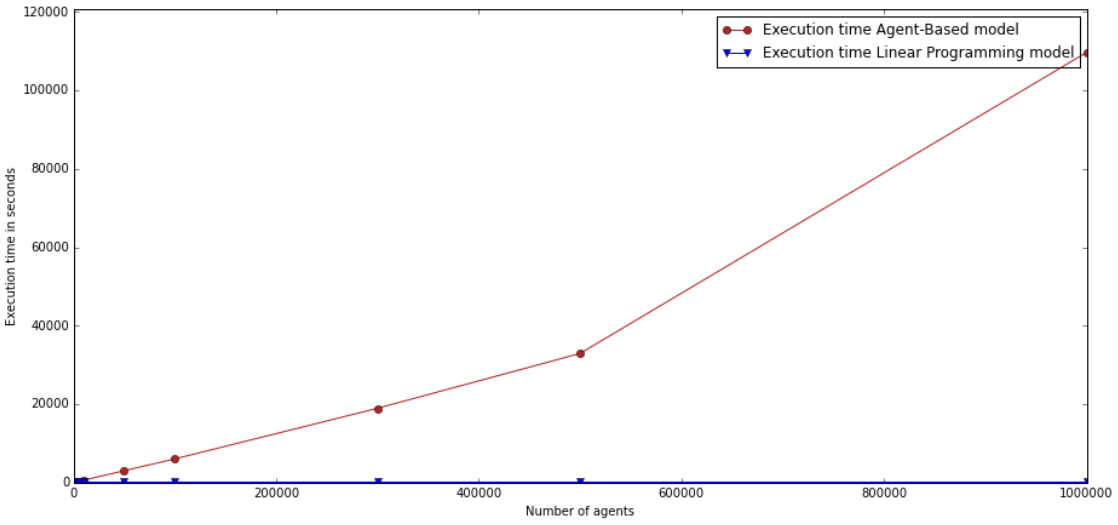


FIGURE 4.9: Compare Linear Programming model And Agent-Based model

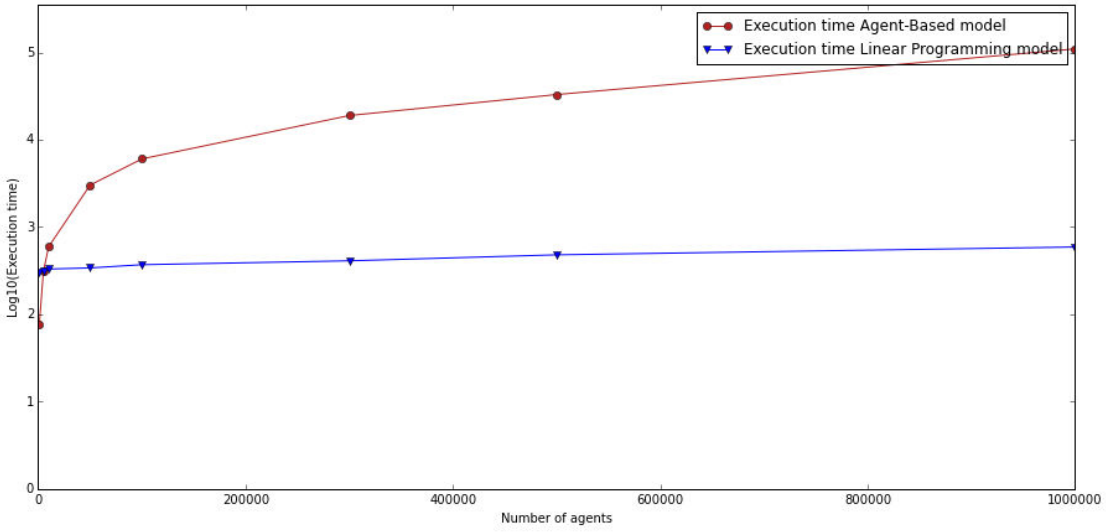


FIGURE 4.10: Compare Linear Programming model And Agent-Based model

execution time of Linear Programming model is stably small in comparison with that of the Agent-Based model. The line representing Linear Programming model is very close to the horizontal axis. On the other hand, the execution time of Agent-Based model grows significantly with the numbers of agents greater than 100000 (hundred thousand agents). Since the curve representing the result from Linear Programming model is so near the x-Axis that it can not be distinguished with the x-Axis, we make another figure (the figure 4.10) in which we represent the log-scale of execution time on the y-Axis. Both figures draw us to the first conclusion that the surrogate model is faster than the Agent-Based model.

Regarding how fast the surrogate is, we focus on the most realistic case of Agent-Based model (the case in which the number of agents is one million and the graph contains

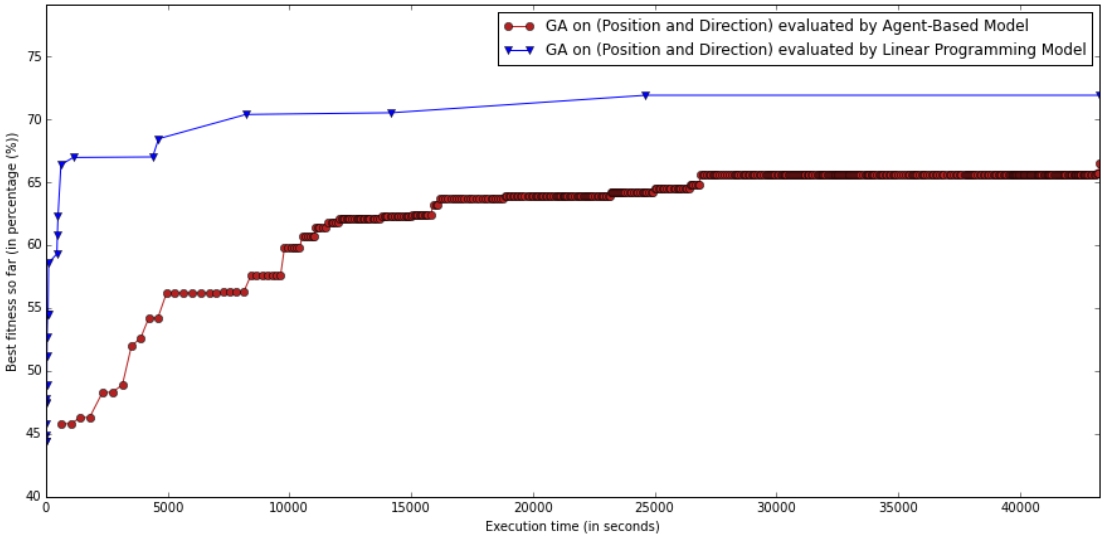


FIGURE 4.11: Compare Linear Programming model And Agent-Based model in Genetic Algorithm

3452 vertices, 10452 arcs). The results from both models show that the surrogate model is 600 times as fast as Agent-Based model.

4.5.4 Comparing performance two models in Genetic Algorithm

Since the application of our approach is to optimizing the deployment of signs, we applied the same genetic algorithm (same kind of evolutionary algorithm with the same coefficient of mutation and crossover) on both implementations. This section presents the comparison of both models in Genetic Algorithm.

Before going to the details of the experiment, we emphasize that the Agent-Based model is too slow to run the Genetic Algorithm with the realistic parameters (e.g number of agents corresponding to real population size, big graph representing the real map of the city). In this case, we build the Agent-Based model as the same as the model in experiment in Chapter 3, section 5 (called experiment 3.5). The graph of experiment representing a small ward of the city (figure 3.5) which contains 65 vertices and 184 arcs. The evacuation time of both models is set 900 steps corresponding to 15 minutes in the real world. The most important factor of this experiment is the termination condition of Genetic Algorithm. In this case, the termination is the execution time which is set 12 hours.

The figure 4.11 presents the result of two implementations of Genetic Algorithm in which the fitness is evaluated by different models:

1. Agent-Based model: this result is acquired from experiment 3.5 which is described in the legend of the chart (called "GA on (Position and Direction) evaluated by Agent-Based Model")

2. Surrogate model: The surrogate model here is the Linear Programming model based on Markov chain decision. In the chart, the legend of this result is called "GA on (Position and Direction) evaluated by Linear Programming Model"

The legends in the chart are a little long in order to clearly distinguish the results of two methods of optimization in this comparison. In fact, there are more comparisons like this in next chapters.

Regarding to the performance of two methods in Genetic Algorithm, the experiment shows the result of the Linear Programming model prevails over the Agent-Based model. The reason is that the linear programming evaluation is much more faster than the agent-based one, then, the genetic algorithm passes more generations. We also applied the best parameters (the list of arcs where the signs are deployed) found by our proposed Linear Programming approach into the agent-based model, and then found that the output result (the percentage of survivors) was also the best, which once again proved the consistency between our proposed Liner Programming surrogate model and Agent-Based model.

4.6 Conclusion

4.6.1 Acceleration of optimization by Linear Programming surrogate model

The first conclusion is that we propose an extremely fast surrogate model replacing Agent-Based model of evacuation which produces nearly the same results of that of the Agent-Based model. The most important factor for the advantage of surrogate model is that the decisions of *simple-behavior* pedestrians are represented into Markov chain decisions. The Agent-Based model of evacuation model of pedestrians is re-modeled into a Linear Programming formulation which is based on Markov chain process.

Practically, both models are tested and compared to each other. The experiments show that the faster surrogate model significantly contributes to improvement of optimization performance. Thanks to surrogate model, the problem of optimizing sign placement in order to guide people in tsunami evacuation becomes practically solvable.

4.6.2 Subproblems to solve

At the end of this chapter, our proposed method opens three subproblems to solve in the next chapters:

1. Complex model of evacuation: This is true that the behaviors of pedestrians are very simple, which opens a question if the surrogate model is still correct in the complex Agent-Based model.

2. Improvement of optimization: What is proposed in this chapter is the improvement of the fitness evaluation. In stead of using costly-computational-effort Agent-Based model, the fitness is evaluated by a very fast Linear Programming model, which opens a question if there is any other improvement for Genetic Algorithm itself.

All these subproblems are solved in the next chapters.

Chapter 5

Decomposition of optimization problem to improve speed

5.1 Introduction

In the previous chapter, we propose a surrogate model which replaces the time-consuming Agent-Based model to evaluate very fast the fitness function. This chapter will focus on how to encode the chromosome of Genetic Algorithm in order to reach the optimal solution faster. Before going to details, we present two issues of our previous implementation of Genetic Algorithm and the idea to overcome these issues.

5.1.1 Large search space

In the previous chapter, we encode a sign placement into a chromosome by using both sign location and sign direction, which means that a sign is represented by an arc (composed of a source vertex and a destination vertex). This representation of chromosome brings the optimization to a very large searching space because the number of arcs is usually greater than number of vertices in the graph representing the city map. On observation of the map of our study case, we focus on two major characteristics of the graph:

1. The number of arcs is much greater than that of vertices: For our experiment, the number of arcs is 3 times as many as that of vertices in 2 scales of the experiment. One represents the full map of the city; the other is only a ward.

2. The number of the candidate directions of a sign is limited: In fact, the degree of every vertex of the graph in our experiment is less than 8. This means that a

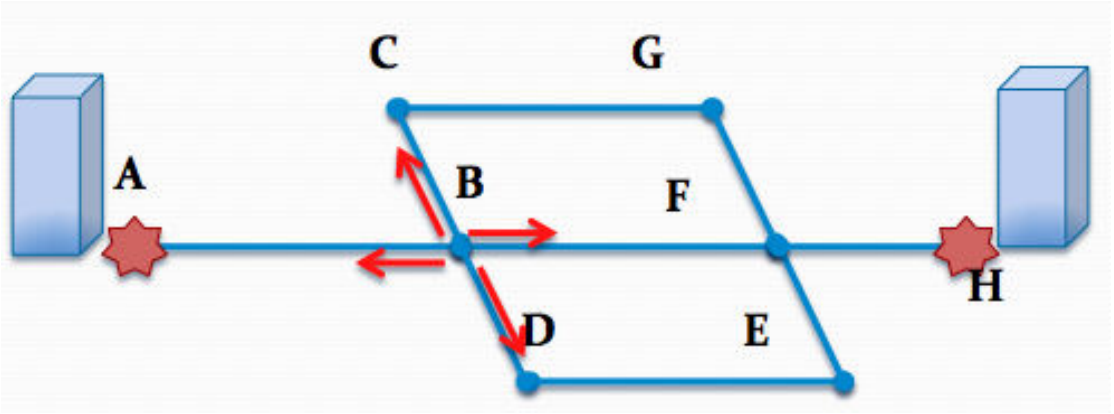


FIGURE 5.1: An example limited choices of directions for a single sign position

vertex does not have more than 7 neighbors. Regarding the representation of a sign, if we fix the location of that sign, the direction candidates is limited. The figure 5.1 shows an example that, if we choose to place a sign at vertex B, there are only 4 directions to which it points.

Thus, in order to reduce the searching space for optimization algorithm, it is reasonable to use only positions of signs to construct the chromosome. The only sub-problem for this approach is how to represent the sign directions.

5.1.2 Unexpected cycles preventing convergence

This representation using both sign positions and sign directions leads the optimization phase falling into cycles. We propose a simple example in figure 5.2 in order to show the problem of the unexpected cycle. At generation i, we have 2 solutions (where to place 4 signs) coded into 2 genomes (CB, FG, BF, ED) and (FE, DB, GC, BF). After the crossover, we have 2 offspring genomes (CB, FG, BF, GC) and (BF, ED, FE, DB) in generation (i+1). These 2 new solutions make the signs point to one another and cause the cycle. Once evacuees arrive any vertices of the cycle, they get lost. When we run genetic algorithm, we always expect that the next generation is better (than the parents). But in this case, the next generation is worse, which makes the optimization phase far from convergence.

5.1.3 Idea of decomposition of problem

Regarding the nature of a sign, a sign always has 2 parts: a position and a direction. Then, the problem of optimizing sign placements can be separated into two parts, which is another aspect of this problem:

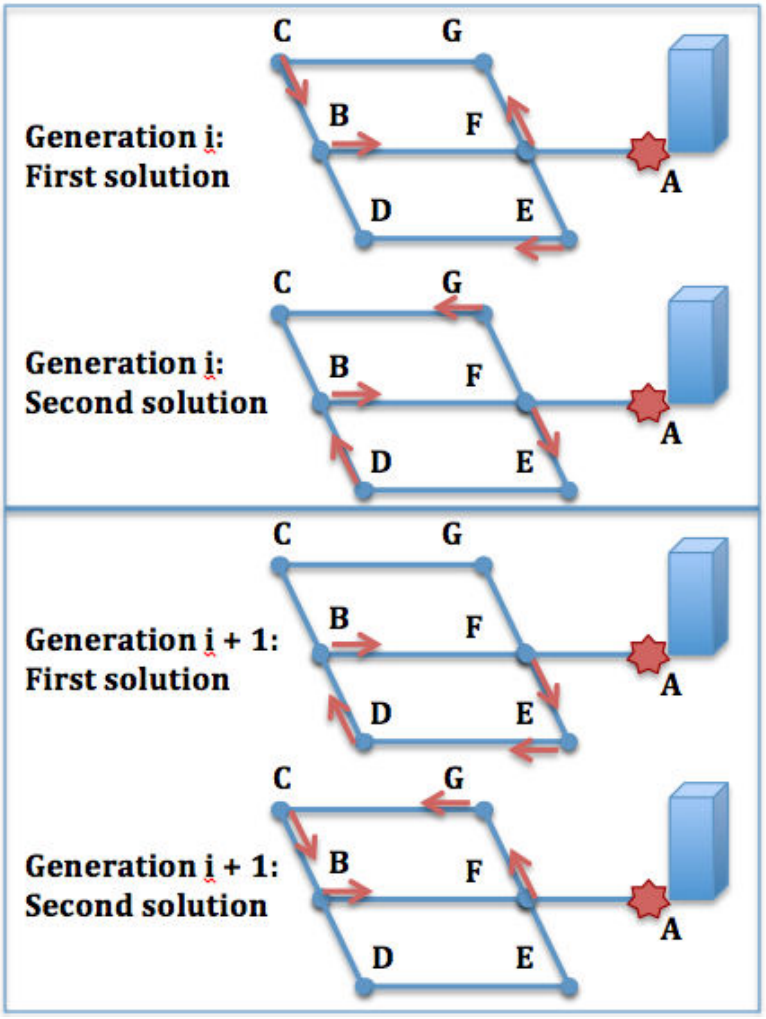


FIGURE 5.2: An example of cycle when running genetic algorithm on both sign positions and sign directions

1. Position part: optimizing the sign positions. It is the harder part because the optimization phase much try to evaluate a great number of position candidatures.

2. Direction part: optimizing the sign directions of each sign. It is the easier part because the number of directions of any sign is limited.

Since this optimization problem can be decomposed into two parts, it becomes a decomposable problem. The most efficient way to solve this type of problem is to focus on solving one part, the other part is computed from this part. Which means that we optimize only the sign positions, the sign directions are computed from these positions. More specifically, the chromosome of this case is encode by a list of vertices representing positions of signs and then the direction of each sign is computed from its position. The important content of this chapter is how to compute the sign directions from their fixed positions such that:

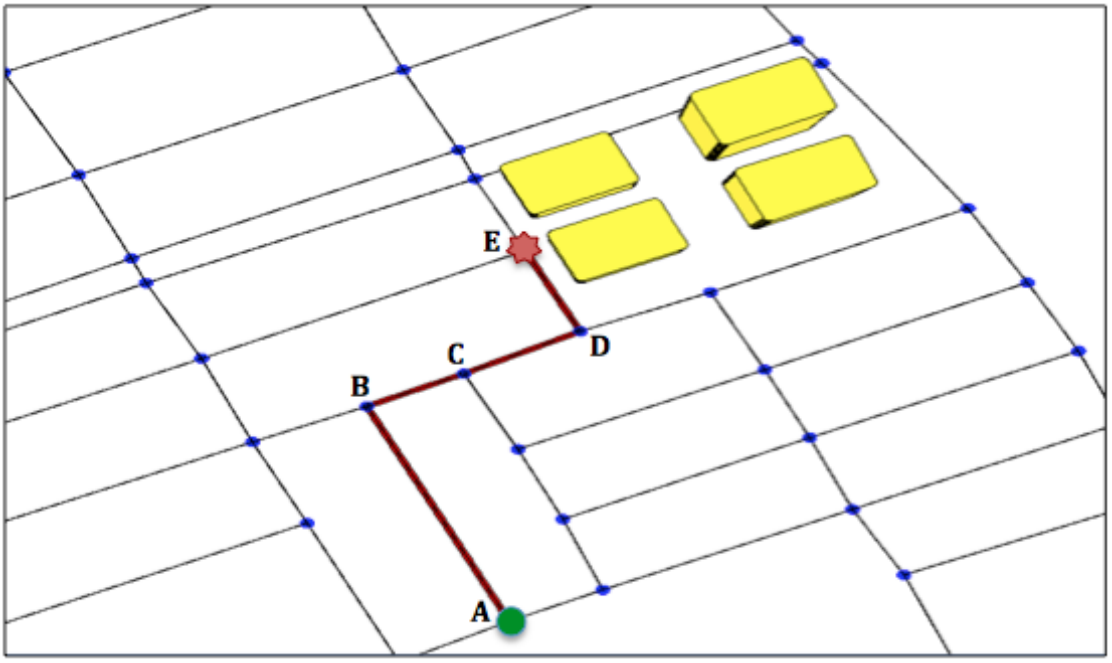


FIGURE 5.3: An example of a shortest path to nearest shelter

- the chosen direction of each sign does not contribute to form any cycles.
- the chosen direction of each sign is the optimal, which maximizes survival rate.

5.2 Precompute direction by shortest path to nearest shelter

5.2.1 Motivation

Psychologically, the most important purpose of every evacuee is to reach the safe place as soon as possible. In order to achieve this purpose, it is reasonable that the evacuee chooses the shortest path to the nearest shelter.

The direction of each sign is approximated by taking the shortest path to the nearest shelter. For an example that we want to place a sign at vertex A (in the figure 5.3), The path (A, B, C, D, E) is the shortest path to the nearest shelter. Thus, for the sign situating at A, its direction should point to B. In our approach, the direction of each sign is independent from any others, we then calculate initially the directions only one time for all the candidate vertices and then apply them to all the experiment of one simulation. Or, we can say that this approximation phase does not slow down the optimization.

5.2.2 Implementation

In this implementation, we re-use the experimentation in the chapter 4, section 5.4 (called experiment 4.5.4). In this experiment, the evacuation environment is a small map of the ward because the Genetic Algorithm needs many evaluations of sign placements. The evacuation time is limited to 15 minutes (as described in the previous chapter). The only difference is the way how we represent chromosomes. The optimization of sign placement is described by following steps:

1. Pre-computation of direction: For each vertex in vertices set, we first compute the shortest path to its nearest shelter. The direction of each vertex is the second vertex on its shortest path. At the end of this step, each sign position has one approximately optimal directions. The figure 7.2 shows an example of pre-computed directions of all the vertices.

2. Re-representation of chromosome: The chromosomes are re-encoded to be a set of vertices which contains exactly K values (corresponding to K signs). These values represent the sign positions (or the vertices where the signs are placed). The figure 7.3) shows an example of a sign placement of two signs (K = 2). In this case, the representation of the chromosome becomes a set of two values (in this example, {1,3}), the directions of these two signs are immediately obtained from the pre-computed directions.

3. Fitness evaluation: The sign directions are directly referenced from the results of step 1. The obtained directions along with the position from encoded chromosomes are evaluated by Linear Programming model (Present in chapter 4)

4. Running Genetic Algorithm with other default parameters: The recombination and the selection are re-used from the previous experimentation. All other parameters are fixed as default values.

5.2.3 Evaluation and discussion

It is usual that when an new optimization method is proposed, it is compared with the previous ones in order to evaluate how good it is. In this evaluation, we compare this method with two previous ones. The three methods (with the same parameters and the same scenario) which are compared here include:

1. GA on (Position and Direction) evaluated by Agent-Based Model: this result is acquired from experiment 3.5

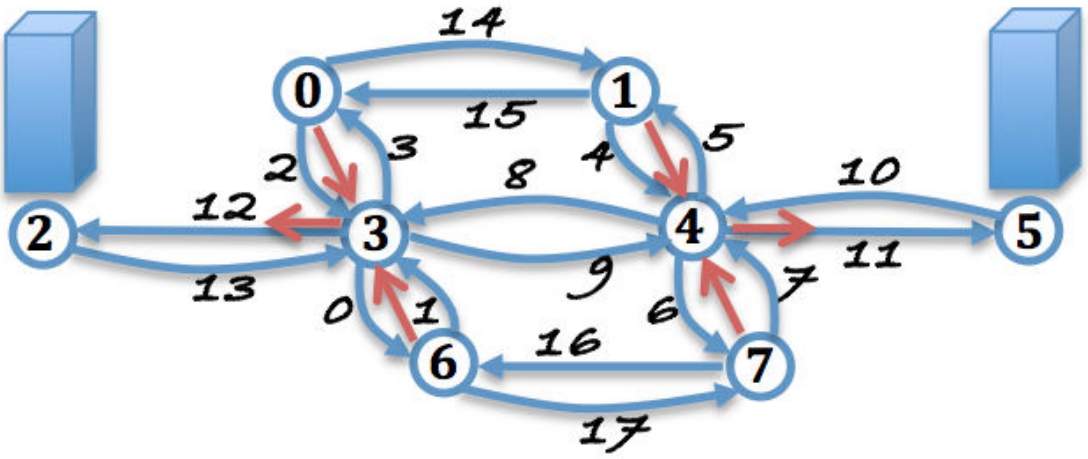


FIGURE 5.4: Example of pre-computed direction on the shortest path to nearest shelter

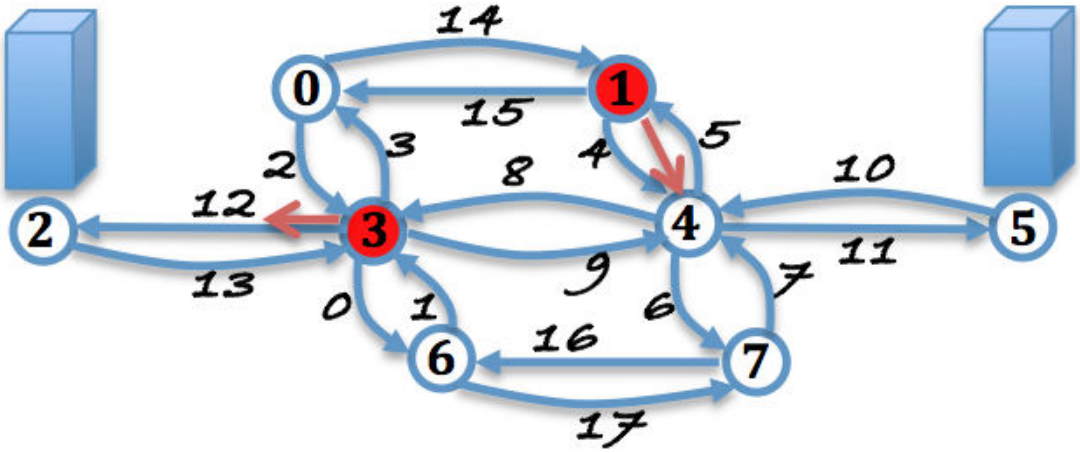


FIGURE 5.5: Example of two signs in pre-computed direction approach

2. GA on (Position and Direction) evaluated by Linear Programming Model: this result is acquired from experiment 4.5

3. GA on Position pre-computed Direction by shortest path to nearest shelter evaluated by Linear Programming Model

The figure 5.6 shows that the new proposed methods (GA on Position pre-computed Direction by shortest path to nearest shelter evaluated by Linear Programming Model) is much better than two others. It is easy to explain its advantage that the searching space is much reduced because of the new representation of chromosome. Moreover, since the directions in this case is pre-computed, which means that the direction of every sign is computed before running the Genetic Algorithm, the execution time of optimization phase is not increased.

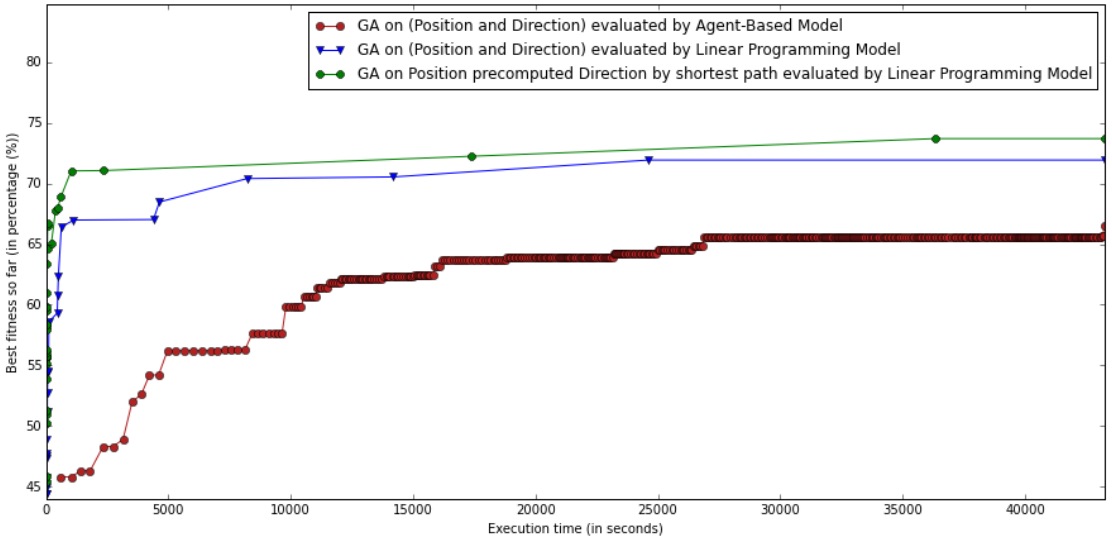


FIGURE 5.6: Result of Genetic Algorithm on only

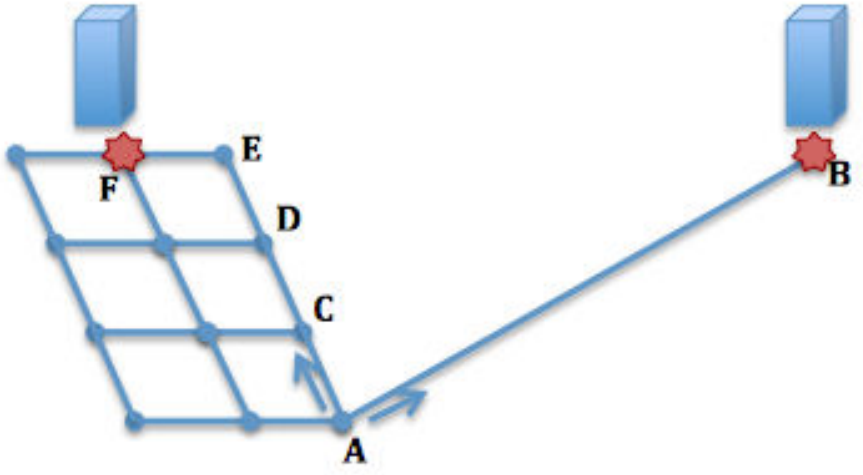


FIGURE 5.7: Lost on complicated shortest path

5.3 Approximate directions by linear programming

5.3.1 Motivation

While the pre-computed direction by shortest path to nearest shelter avoids the cycles, it has another problem called "lost on complicated shortest path". In fact, the complex shortest path is worse than a simple longer path in some cases. In figure 5.7, the path (A, C, D, E, F) is the shortest path but very complicated. If evacuees follow this path, they might get lost at the next "no signed" junctions. Then, for a sign locating at A, it should point to B (longer but very simple path), but in the recent work the direction was proposed to point to C, which causes the problem.

According to the pre-computation of the shortest path to the nearest shelter, a direction of a sign position does not depend on any other signs. This slightly contradicts to the fact that the group of successive signs can improve the survival rate. In order to benefit the successive signs, it would be better to use another approach which approximates the sign directions by taking account of other signs.

As we mention in the chapter 3, in NGUYEN et al. [2011], the authors presented the Linear Programming approach that Minimize the Average Evacuation Time (called MAET). While this MAET approach does not address to the number of survivors, it opens a way to optimize the direction from the predefined position. In stead of using this approach to find the optimal sign placement (including sign positions and also sign directions), we use it to find the directions for the fixed positions of signs such that the Average Evacuation Time is minimum. In order to use the MAET to our purpose, we re-model the original Linear Programming formulation of MAET (described in NGUYEN et al. [2011]).

5.3.2 Approximate directions in order to Minimized Average Evacuation Time

First we proposed some definitions to clarify the formulation. Let q_i denote the Average Evacuation Time for pedestrians beginning at vertex i and the variable $a_{ij} \in \{0, 1\}$ indicates whether a sign was placed on edge $\{i, j\}$

In our problem, we suppose to have a predefined position of signs. Let S $\subset$ V denote the set of vertices where we place the signs. We then reuse this formulation but also propose to add two other constraints to optimize the direction from the fixed positions.

$$\forall i \notin S,\ a_{ij} = 0 \tag{5.1}$$

$$\forall i \in S,\ \sum_{j \in N(i)} a_{ij} = 1 \tag{5.2}$$

Thus, the Linear Programming formulation for optimization of Sign Directions from predefined Positions becomes:

$$
\begin{aligned}
\min \sum_i \mu_i.q_i \qquad & \textit{such that} \\
\forall i \in X,\ q_i &= 0 \\
\forall (i,j) \in A,\ q_i &= c_{ij} + q_j \text{ if } a_{ij} = 1 \\
\forall i \in V,\ q_i &= \sum_{j \in N(i)} p_{ij}.\{c_{ij} + q_j\} \text{ if } \sum_{j \in N(i)} a_{ij} = 0 \\
\forall i \notin S,\ a_{ij} &= 0 \\
\forall i \in S,\ \sum_{j \in N(i)} a_{ij} &= 1 \\
\sum_{(i,j) \in E} a_{ij} &\leq k
\end{aligned}
$$

5.3.3 Implementation

In this implementation, we re-use the experimentation in the chapter 4, section 5.4 (called experiment 4.5.4). In fact, all the experiments on Genetic Algorithm explore the same model of only one scenario of evacuation. In this scenario, the evacuation environment is a small map and the evacuation time is limited to 15 minutes. The optimization of sign placement is described by following steps:

1. Re-representation of chromosome: The chromosomes are re-encoded to be a set of vertices which contains exactly K values (corresponding K signs). These values represent the sign positions (or the vertices where the signs are placed).
2. Fitness evaluation: The fitness function in this case passed two steps:
   (a) Approximate sign directions by Linear Programming with MAET approach (previous formulation)
   (b) Calculate survival rate by Linear Programming based on Markov chain decision (described in chapter 4)
3. Running Genetic Algorithm with other default parameters: The recombination and the selection are re-used from the previous experimentation. All other parameters are fixed as default values.

5.3.4 Evaluation and discussion

At this step, we evaluate the new proposed method by comparing all the optimization methods. The 4 optimization methods include:

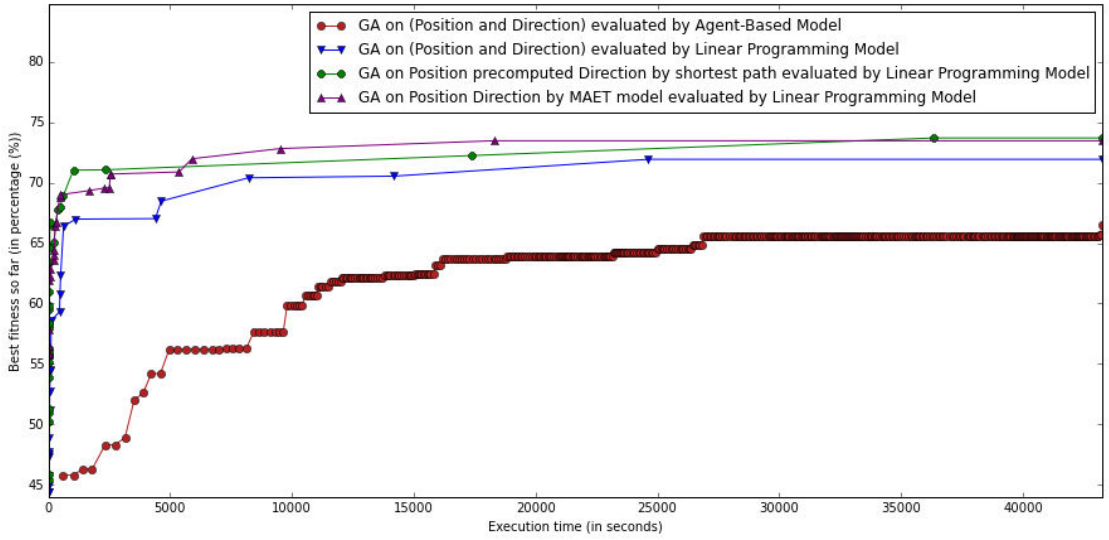


FIGURE 5.8: Compare all optimization methods

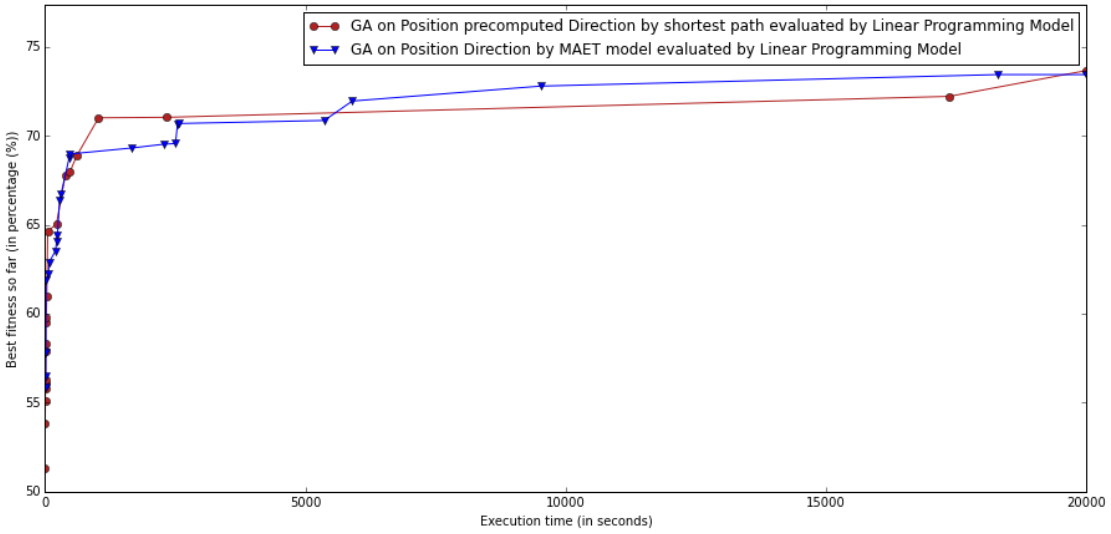


FIGURE 5.9: Compare pre-computed direction by shortest path and approximated direction by MAET

1. GA on (Position and Direction) evaluated by Agent-Based Model: this result is acquired from experiment 3.5

2. GA on (Position and Direction) evaluated by Linear Programming Model: this result is acquired from experiment 4.5

3. GA on Position pre-computed Direction by shortest path to nearest shelter evaluated by Linear Programming Model

4. GA on Position approximated Direction by MEAT evaluated by Linear Programming Model

In the figure 5.8, the two new proposed methods are the best. The two reasons for these results are that:

1. New representation of chromosome significantly reduces searching space.
2. New representation eliminates the unexpected cycles, which prevents the optimization phase from reaching convergence.

These two best methods are compared to each other in figure 5.9. According to experimental results, it is hard to conclude which is the better method between the two ones. The only explanation for this fact is that each method has its own pros and cons. While the "Shortest path direction" has problem with the shortest complicated path, the MEAT does not address to survival rate.

5.4 Conclusion

5.4.1 Solving decomposable problem with pre-computation

In this chapter, we propose another aspect of the problem of optimizing sign placements. From this aspect, the problem becomes decomposable into two parts: one part concerning the sign position, the other relating to sign direction. In order to benefit this aspect, we propose a new representation of chromosome of the Genetic Algorithm which is used to solve this optimization problem. The new representation only encodes the positions of signs. The sign directions are computed from these sign positions.

We also propose two methods to compute the sign direction, which leads to new optimization methods to solve the optimization problem of sign placement.

1. GA on Position pre-computed Direction by shortest path to nearest shelter evaluated by Linear Programming Model
2. GA on Position approximated Direction by MEAT evaluated by Linear Programming Model

While none of these new proposed methods prevails others, they are both better than the previously proposed methods.

Thanks to these propositions, the optimization of sign placement in which the survival rate is evaluated by a simple-behavior pedestrian model becomes tractable.

5.4.2 Subproblems to be solved

According to propositions in the previous chapters, the optimization of sign placement is tractable. However, the evacuation model used to evaluate a sign placement is very simple. It is true that more complex evacuation model would make the optimal result more convincing. But, the complex model needs much more computational resources to evaluate the sign placement, which makes the optimization problem becomes infeasible with respect to execution time.

Chapter 6

Adapting the Linear Programming approach to more Complex Agents

6.1 Introduction

In the previous chapter, while the proposed method is tractable with respect to execution time, it poses a question about the practicality of the evacuation model. The behaviors of evacuees in this model are too simple that it is hard to be accepted in the real world application. In the model, there is only one type of agents representing the tourists who do not know the map. The other type of agents such as the citizens who absolutely know the map are not mentioned in the simulation. Moreover, there are neither crowd-based behaviors nor human interaction in the evacuation which are needed in the real evacuation.

In this chapter, we propose an approach to deal with complex-behavior model of evacuation. This approach is, in fact, an approximation method which approximates a complex Agent-Based model into a Linear Programming mode based on Markov chain decision. Precisely, our goal is to mimic as well as possible the behavior of complex agent models, using simple models such as Markov chains. We will then be able to compute the survival rate of these simple agents using the linear programs of earlier chapters. To build the stochastic transition matrix of our simple agents, we will:

1. run simulations using complex agent model.
2. record all movements of the agents during these simulations.

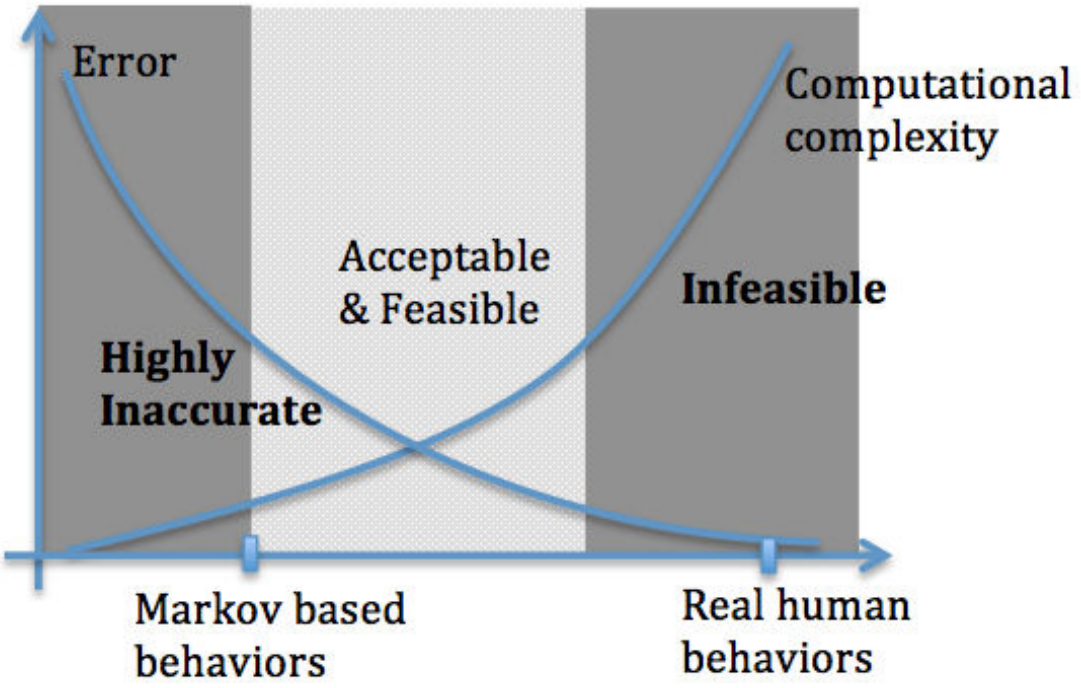


FIGURE 6.1: Trade off between accuracy and computational speed

3. build a stochastic transition matrix P and cost matrix C such that:
   - $P_{i,j}$ refers to the proportion of agents in the simulation which chose to cross the arc (i,j) when they were located on vertex i
   - $C_{i,j}$ refers to the average time of agents which cross arc (i,j)
4. build simple agents, whose behavior will be to follow signs if there are some, otherwise to move from vertex to vertex, following edges randomly, according to this transition matrix.

Before going to details, we present the trade off between the speed and accuracy in the optimization. The trade off between the speed and accuracy is a regular relation in the reality (an example presented by WICKELGREN [1977]) as well as in computer science (a case study described by BOUSQUET and BOTTOU [2008]). This is quite similar to the philosophic relationship between quantity and quality. Our optimization problem is not exceptional. The figure 6.1 illustrates this trade off in our problem. If we built a simulation in which the agent behaviors are very close to the real human behaviors, the problem becomes infeasible because of its computational complexity. However, if we use an over-simplified simulation (e.g. Markov based decision), the accuracy is unacceptable. In order to balance these two objectives, we propose to build a model whose accuracy is acceptable and whose speed is as fast as an over-simplified simulation (like Markov chain

decision model). Next section of this chapter presents a complex model of evacuation which includes 3 realistic factors of an evacuation model:

1. Crowd-based behaviors: In this model, evacuees tends to form the groups.
2. Communication: The evacuees communicate to share the evacuation paths.
3. Traffic jam: The more evacuees in a narrow road would reduce the evacuation speed.

6.2 Complex Agent-Based model of evacuation in case of tsunami

In fact, the human decision is much more complex than Markov Chain process. In a real evacuation, people can communicate each others. They can share information about where to go because there are always parts of people (e.g. the citizens, the police men) who do know the map of the city. Moreover, the high density of evacuees in the narrow street might reduce the evacuation speed.

6.2.1 Leader/follower behaviors in crowd-model

Since the evacuation from tsunami is a very urgent situation, the evacuees behaviors are obviously complicated. However, there is a common social phenomenon of evacuation that people usually evacuate in groups or at least the movement of people can form the group. We believe that people who do not know the map would try to get it or, at least, follow another who knows it. The fact that people follow each other in evacuation forms the groups.

According to our survey on the related works, the way that some evacuees follow others to form the group has received much research in recent years. While some studies called this phenomenon the leader/follower model (studies proposed by JI and GAO [2007], PELECHANO and N.I. [2006]), others used another name "sheep and fox" model (presented by NGUYEN et al. [2012d]). No matter what the phenomenon is named, there are always 2 types of evacuees in the evacuation: one knows the map (called leader, or fox), other does not know (call follower, or sheep).

In our work, we also embed the leader/follower behaviors into evacuation simulation in order to evaluate the sign placement because leader/follower behaviors are generic enough to describe these social behaviors. The Agent-Based simulation of leader/follower evacuation here has 2 types of agents:

1. Leader: The agents representing people who know the map. These agents take the shortest path to the nearest shelter. And, during the movement, they spread the information to nearby agents. These agents can ignore the signs if the directions conflict with their knowledge.

2. Follower: The agents who do not know the map. If these agents perceive any "Leader", they follow that "Leader". Otherwise, they carry out the basic decision: follow the sign if they perceive it or take turns uniformly otherwise. Precisely, a follower perceives a leader only if both are within a give distance. This distance is called communication range and is set as 10 meters (in our experimentation).

Once a leader agent is within the range of communication of any follower agents, the communication happens. What the leader has is the shortest path to the nearest shelter, which is also what the follower wants. The exchanged information then is the shortest path to the nearest shelter. Once a follower holds the information of the shortest path, he can play a role as a leader by sharing this information to other followers on the routes.

In this model, the signs are dominated by the leader agents, which means that if a sign and a leader agent point to different directions, follower agents follow the leader and ignore the sign. It is also true in the reality that in case the traffic light and the policeman provide different signals (e.g the traffic light is still red but the policeman permits people to pass), people should follow the policeman. Then, in our model, the effect of the signs might be reduced if their directions are not well oriented. The effect of well oriented signs is discussed in the experiment section.

6.2.2 Agent's speed and density

It is true that the more crowded the street is, the slower the evacuation speed becomes. As an agent is represented by a point, it is obligatory to compute the speed according to the density. In this case we reuse the formula proposed in GOTO et al. [2012] (illustrated in figure 6.2) which are described as follow:

1. An agent has a maximum speed.
   (a) For a normal family, the value of maximum speed is 1.5 m/s
   (b) For a family having infants or old peoples, the value is 0.75 m/s
2. Regarding to density which reduces the speed of a specific agent, we consider the number of other agents in from of the agent itself. Let that number of agent to be #Front.

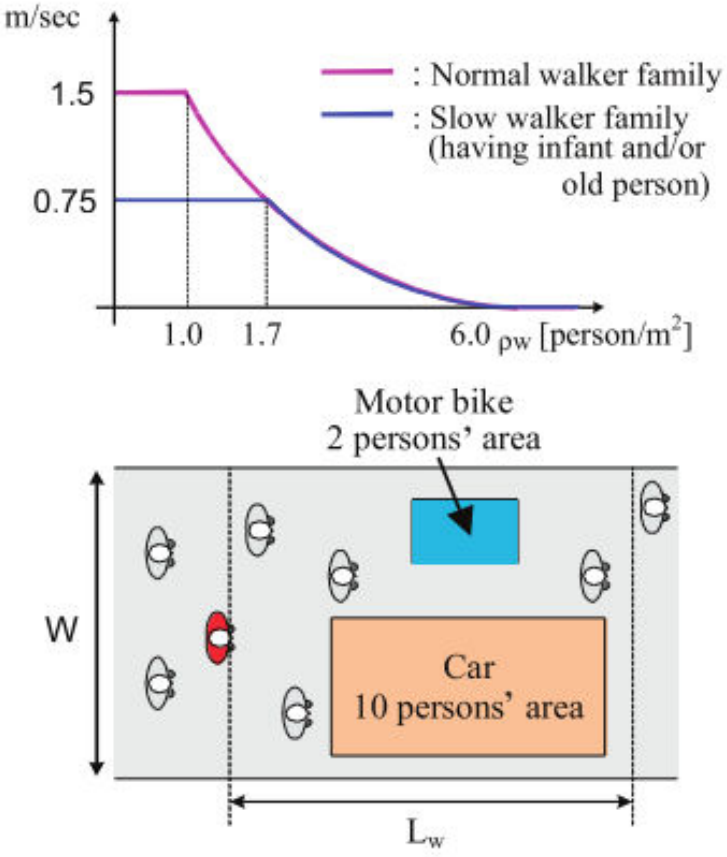


FIGURE 6.2: Relationship between speed and density proposed in GOTO et al. [2012]

3. The area of consideration (S) is computed by multiplying watching distance ($L_w = 3m$) with road width (W) $S = W \times L_w$

4. The density is normally computed by the division of number of agents over the area: $D = \#Front/(W \times L_w)$

5. The speed of an agent begins to slow down from the moment that the density in front of the agent itself is 1.0 $person/m^2$ and continues increasing.

6. The speed is zero when the density reaches to 6.0 $person/m^2$

6.3 Initial approximation of complex model to simple model

As we mention in the chapter 4, the probabilities transition matrix (P) and the cost matrix (C) are the most important informations for Markov chain. Then, the purpose of the approximation is to build these matrices so that the decision (in this case, the agent

decision to turn into a certain direction from a crossroad) given by Markov chain behaviors is similar to that of leader/follower behaviors and that the cost matrix represents the average time requiring an agent to move between all pairs of vertices.

The idea of this approximation method is that we note how many times agents turn from a crossroad to another and how many steps it takes an agent to go from a crossroad to one of its neighbor.

6.3.1 Probability transition matrix (P)

Practically, we organize a temporary matrix TP ($|V| \times |V|$) in which the value of TP_{ij} represents the number of times that agents turn from vertex i to vertex j. Initially, $TP_{ij} = 0, \forall i, j \in V$. Once an agent makes decision to turn from vertex i to vertex j, the value TP_{ij} is updated by increasing by 1. After repeating the simulation 1000 times, we compute the probabilities transition matrix (P) from the value of matrix TP.

6.3.2 Cost matrix (C)

The cost matrix represents the time for an agent to move from vertex to vertex. At the beginning of the simulation, every agent is supposed to be at a certain vertex and this agent moves from vertex to vertex until it reaches a shelter. At a certain moment (t_i) of the simulation, an agent begins at vertex i and intends to move to vertex j. Once this agent arrives at vertex j (at the moment t_j), the cost for that agent to move on the arc (i,j) is $c_{ij} = t_j - t_i$. In this case, there are plenty of agents. The c_{ij} has the average value of all the agents passing the arc (i,j).

In the implementation, we organize 2 temporary matrices:

1. Matrix TC($|V| \times |V|$) stores the accumulating evacuation time of all the arcs. Initially, $TC_{ij} = 0, \forall i, j \in V$. Once an agent beginning at vertex i arrives at vertex j, the value $c_{ij} = t_j - t_i$ is updated by accumulating the evacuation for this agent to go from i to j.

2. Matrix Count($|V| \times |V|$) counts the time that an arc (i,j) is passed.

After finishing the approximation phase (1000 times of repeating simulations), the cost matrix C contains the values $c_{ij} = TC_{ij}/Count_{ij}$.

6.3.3 Strategy of approximation and evaluation

As we mention in the Markov chain description in chapter 4, the probability transition matrix and the cost matrix are predefined before adding the sign placement into the Linear Programming model. These 2 matrices are independent from the sign placement. More precisely, the Linear Programming model is able to compute the survival rate even when no sign is placed. Following this idea, the approximation phase focus on the case with no signs. The strategy is described as follows:

1. Approximation: We run Agent-Based model of leader/follower with no sign.
   - The agents' decisions are noted to build probability transition matrix
   - The evacuation time of every road is noted to build cost matrix
2. Using approximation result in Linear Programming model: The 2 matrices are given to Linear Programming model in order to compute the survival rate
3. Evaluation of approximation: The purpose of this step is to show if the result from the 2 models are similar.
4. Comparison of optimization method: The 2 models are used as the fitness function in 2 different implementations of Genetic Algorithm.

6.4 Implementation

First, the scenario is this case is a short-time evacuation which indicates that the evacuation time is 15 minutes (corresponding to 900 seconds). The evacuation map is the small map of a ward of the city. This scenario is the same as that of the implementation in the previous chapter. Basing on this scenario, we build 4 implementations: Agent-Based model of complex crowd behaviors, Linear Programming model of approximating behaviors and 2 implementation of Genetic Algorithm (one uses Agent-Based model to evaluate fitness, other uses Approximated Linear Programming Model)

6.4.1 Agent-Based model of complex behaviors of pedestrians

From the description of evacuee behaviors in the previous section, the actions of each evacuee are described as follows:

1. If the evacuee is a leader (the one who knows the map), he is considered as a group himself for he is the group leader.

(a) If the leader perceives any follower in his range of communication (This coefficient is predefined as the value as 10 meters)

i. The leader invites the follower to join the group.

ii. The leader wait for the follower to approach the group.

iii. If the follower is near the leader:

A. The leader share the guiding information to the follower. The information here is the shortest path to the nearest shelter.

B. The follower becomes the leader for he knows the evacuation path.

C. All the leaders move to the nearest shelter on the shortest path that they know.

(b) If the leader does not perceive any followers

i. He evacuates to the nearest shelter.

2. If the evacuee is a follower (the one know does not know the map)

(a) If the follower perceives any leaders in his range of communication

i. The follower approaches the leader.

ii. The follower receives guiding information from the leader.

iii. The follower becomes a leader once he has the shortest path to the nearest shelter.

iv. The follower evacuates to the nearest shelter like a leader.

(b) If the follower does not perceive any leaders in his range of communication

i. If the follower perceives a guiding sign, he follows the sign direction.

ii. If the follower does not perceive any guiding signs, he randomly takes a turn and continue evacuating.

3. At each step of the simulation, the evacuee evaluates the density of people in front of him in order to calibrate his movement speed.

6.4.2 Linear Programming model of approximating behaviors

The Linear Programming model in this case is the same Linear Programming described in chapter 4, excepts the input matrices. The input matrices in this case are the probability transition matrix and the cost matrix which are obtained from approximation phase.

6.4.3 Genetic Algorithm with fitness evaluated by Agent-Based model

This implementation is also the same as the Genetic Algorithm in chapter 3, excepts the fitness function. The fitness function in this case is evaluated by Agent-Based model of complex-behavior pedestrians. The chromosome of the implementation is coded as the list of arcs (using both sign positions and sign directions). Since the fitness function is evaluated by complex model, it takes more time to evaluate it.

6.4.4 Genetic Algorithm with fitness evaluated by Approximated Linear Programming model

In this case, we create 2 different versions of Genetic Algorithm. Each version uses a different representation of chromosomes.

1. Chromosomes are encoded by sign positions and sign directions. This implementation is the same at the implementation in chapter 4, excepts that the fitness function is evaluated by Approximated Linear Programming model.

2. Chromosomes are encoded by only sign positions. Sign directions are pre-computed by the shortest path to the nearest shelter. This is similar to the implementation in chapter 5. The fitness function here is evaluated by the approximated model.

All the models simulate 1000 agents in which there are 10% leaders and 90% followers. The initial distribution of agents is uniform, which means that the probabilities for an agent to be initial at any vertices are equal. All the common coefficients of all the models are fixed as the same values.

6.5 Experiment and evaluation

6.5.1 Verification method for approximation

In this experiment, we run two implementations: the Agent-Based model of complex-behavior pedestrians and the Approximated Linear Programming model. The purpose of this experiment is to evaluate if they produce the similar results. In order to achieve this goal, we first focus on the most important indicator, the percentage of survivors. The evaluation becomes comparison of the survival rate from 2 models. We perform the experiment as following steps:

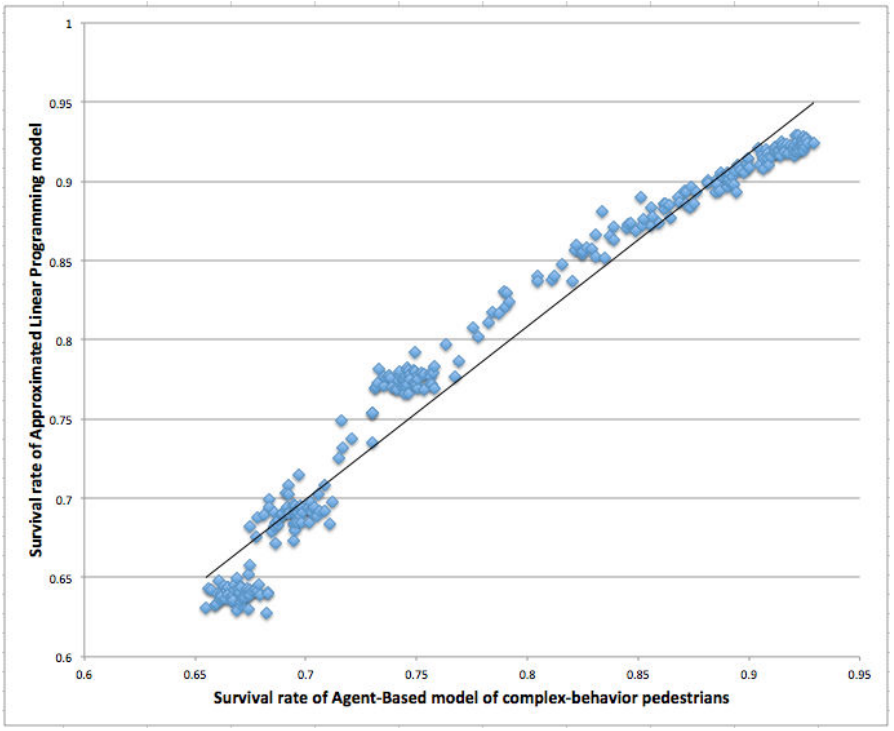


FIGURE 6.3: Random signs are blindly generated

1. We run repeatedly Agent-Based model of leader/follower with no sign. The results of this step are the probability transition matrix and the cost matrix.

2. We generate random signs (random number of signs, random location and random direction)

3. For each sign placement from the previous step, we run 2 models and note the survival rates from these 2 models. The models are:

   (a) Agent-Based model of leader/follower with the input parameter is the sign placement (both sign positions and sign directions).

   (b) Linear Programming model in which: the sign placement is the same as that of the Agent-Based model; the probability transition matrix and the cost matrix come from the first step.

In this experiment, the signs in this case are blindly generated (random number of signs, random locations and random directions). The result of this test case shows that the approximation does not give much error but the variation is quite complicated. In figure 6.3, the dots distribute quite far from the trend line and the form of the distribution is not either very similar to the trend line. The reason for this phenomenon is that: in the leader/follower agent simulation, the leader do not follow the signs if the directions are not good, but in the Linear Programming model, the agent must follow the signs; then,

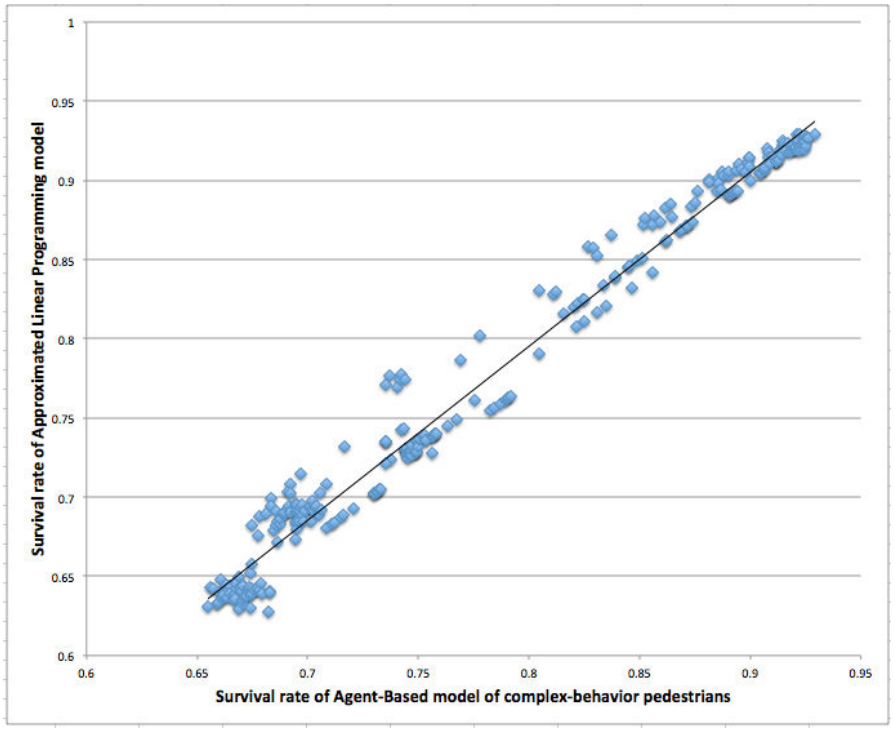


FIGURE 6.4: Increasing number of signs with well oriented

in case of random sign directions, the percentage of survivors of both simulations are usually different.

Since the leaders always use the shortest path to the nearest shelter, they ignore the signs which are not well-oriented. In order to improve the performance of the signs, we propose to make them well-oriented. Then, we propose to use the pre-computation of sign directions using the shortest path to the nearest shelter (proposed in chapter 5), which is similar to leaders' decisions. In order to evaluate this proposition, we re-run the experimentation in which the signs is only represented by their positions, their directions is precomputed by the shortest path to the nearest shelter.

The figure 6.4 presents the result of this version of experiment. When the signs are well oriented, the experiments show that more signs save more people. In the result, the percentage of survivors in Approximated Linear Programming model are linearly similar to that of Leader/follower one. In this figure, the dots distribute around the trend line and become converging when number of signs increase.

The two test cases lead the evaluation to 2 very important conclusions for the sign placement optimization in evacuation simulation:

1. We can approximate a typical social simulation of evacuation to Linear Programming model of Markov chain decision with acceptable error.

2. If the signs are well-oriented, the result of the approximation of Markov decision simulation is very close to that of complex agent-based simulation, which makes the approximation reliable.

6.5.2 Evaluation of approximation in optimization methods

This is the main experiment of this chapter. In this experiment, we find out if the approximation helps the optimization method. In this case, we firstly run the approximation model to get the probability transition matrix and the cost matrix for the Approximated Linear Programming model. Then, we run 3 implementations to get the survival rate.

1. Genetic Algorithm in which chromosomes are represented by sign positions and sign directions, fitness is evaluated by Agent-Based model of complex-behavior pedestrians (called method 1).
2. Genetic Algorithm in which chromosomes are represented by sign positions and sign directions, fitness is evaluated by Approximated Linear Programming model (called method 2).
3. Genetic Algorithm in which chromosomes are only represented by sign positions, sign directions are precomputed by the shortest path to the nearest shelter, fitness is evaluated by Approximated Linear Programming model (called method 3).

In this experiment, the best solutions are re-evaluated by Agent-Based model of complex-behavior pedestrians in order to maintain the reliability. More concretely, the solutions of optimization methods are the sign placements. These sign placements are given to Agent-Based model to re-compute the survival rates which are used to compare the performance of optimization methods.

The figure 6.5 shows the comparison of 3 methods. For the first method where the fitness is evaluated by the Agent-Based model, the solution gradually gets better over the time. The curve representing method 1 grows during the execution time. This is quite contradict to 2 other methods where fitness is evaluated by the Linear Programming model. On observation from the chart, the curves representing these two methods (where fitness is evaluated by Linear Programming model) are quite zigzag. The reason is the differences between the survival rates given by 2 models (one is Agent-Based model, other is Linear Programming model).

Regarding the performance of the optimization methods, the method 3 is the best among three methods. The curve representing this method is slightly zigzag at the first generations when the solution is far from convergence. Gradually, the survival rate of this

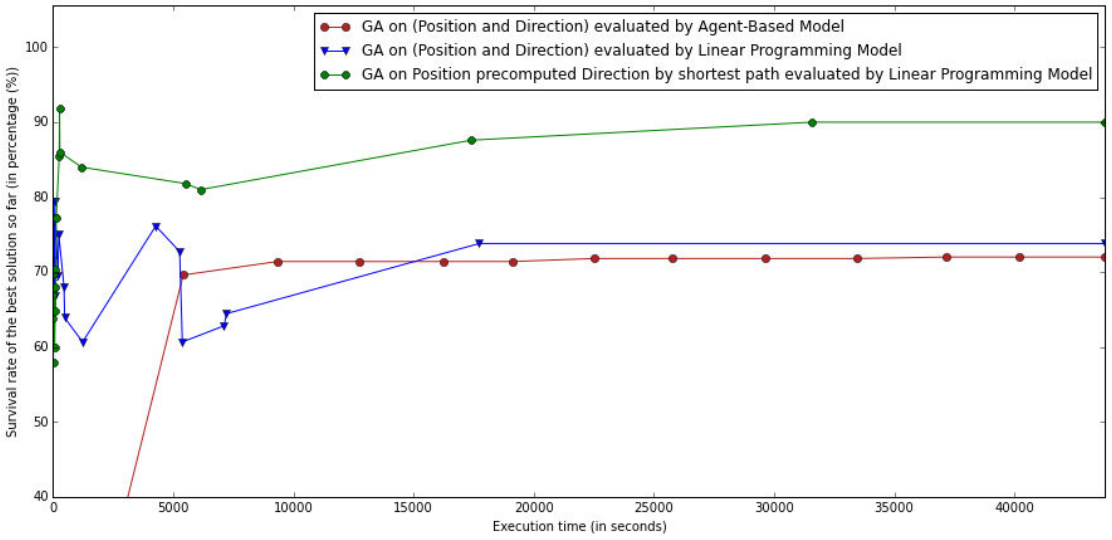


FIGURE 6.5: Comparison of optimization methods with complex-behaviors evacuation

method grows. The reason for this result is the shortest path to the nearest shelter. In the leader/follower model, the shortest path is the behaviors of the major parts of the crowds because more and more followers receive the shortest path thanks to the communication during the evacuation. In the method 2, the shortest path is the direction of a sign, which makes the sign placement more valuable. In fact, the pre-computation of sign directions removes significantly the useless signs which contradict to the movement of the leaders.

6.6 Conclusion

6.6.1 Contribution

In this chapter, we proposed a model of a complex-behavior evacuation. In this model, there are 3 convincing factors of a realistic evacuation: the crowd behaviors, the communications, and the traffic jam caused by the over-crowded density. Regarding the solution to the problem is the optimization of sign placement of the complex evacuation, we propose the approximation method which approximates complex Agent-Based model into Linear Programming model of Markov chain. This approximation makes the problem tractable with respect to execution time. At this moment, all subproblems receive acceptable solutions. The most realistic solution for the complex model is that:

1. The Linear Programming model of Markov chain decision is build in order to evaluate the survival rate.

2. The Agent-Based model of complex-behavior pedestrians is run to learn the pedestrians movement. The result of this step is the probability transition matrix and the cost matrix which are used by Linear Programming model to evaluate the survival rate.

3. Genetic Algorithm in which chromosomes are only represented by sign positions, sign directions are precomputed by the shortest path to the nearest shelter, fitness is evaluated by Approximated Linear Programming model.

4. The best solution of the Genetic Algorithm of every generation is re-evaluated by Agent-Based model.

6.6.2 Future works

We present here the ideas to improve the solution of our study in order to make the solution more realistic.

6.6.2.1 Improvement of Genetic Algorithm

In our implementation of Genetic Algorithm, the crossover and the mutation are the simple operators with the default coefficients. It is needed to try other operators (e.g. the geometric crossover) and also to tune these coefficients in order to accelerate the optimization phase to reach the convergence.

6.6.2.2 Approximating injection

Our proposition of approximation is only the initialization, which means that the approximation happens before the Genetic Algorithm. Although the approximation works in the experiment of a small evacuation (small map, limited number of pedestrians), the accuracy of the approximation is still under discussion in the real evacuation. In order to improve the reliability of the approximation, we propose an idea to inject it into the evaluation process of the Genetic Algorithm. Specifically, after a certain number of generations, the approximation is run in order to update the probability transition matrix and the cost matrix.

6.6.2.3 Profile of the population

In the evacuation, the population is divided into two types: leader, follower. In the real world, the profile of the population is much more complicated. The population can be

categorized into different groups. The individuals of each group might have different behaviors. For an example of the age of the people, the population can be categorized into: children, old people, adults. The behaviors of the children is obviously different to those of the adults. And then, we have plenty of characteristics to categorize the population, such as: sex, age, job, disability condition.

6.6.2.4 Initial distribution of the population

In this study, the distribution of the population is uniform, which means that the probabilities for all agents at the beginning of the evacuation are equal. In the real world, the distribution of population depends on the time. In daylight, most people are at work. In the midnight, most of them are at home. In cold winter, there are few people near the beach. In hot summer, the beach is covered by crowed people. Then, the distribution of people in the real world is much different to uniform distribution. In order to bring the solution to application, the distribution of people depending on certain period of time must be taken into account.

Chapter 7

Predicting the Survival Rate with Regression Methods

7.1 Introduction

In the previous chapters, we propose 2 different methods to evaluate how good a sign placement is. These methods in fact compute the survival rate of a given sign placement. From a placement of K signs (sign position and direction) we evaluate the percentage of survivors (or survival rate). One is Agent-Based model of pedestrian evacuation which is considered highly accurate but too slow to do optimization. The other is called Linear Programming model of Markov chain which is faster but less accurate. However, we find that the speed is not fast enough. We thought that it would be very fast if we map directly from sign placements to survival rates, which motivates to learn this direct mapping.

In this chapter, we propose another method to predict survival rate, which leads to another method of optimization of sign placement. The new method is the prediction of survival rate by regression. First, we run simulation on M examples of random sign placements. We use a "learner" to train these obtained data. Then, we use this learner to predict the survival rate of a new sign placement. Once we have a predictor for survival rate, we will search for the best sign placement with respect to this prediction. This method is expected to be extremely faster than the simulation. However, the main issue of this method is always the problem of accuracy, which is obviously trade-off for this speed. Before going to the details of the prediction, we present how to represent the sign placement, how to build the features.

7.2 Proposition of feature representation

The main purpose of this chapter is how to learn the direct mapping from sign placements to survival rates. However, the "learner" needs the features, which means that we have to represent the features from the input sign placement. This section, we present the different approach to represent the features.

7.2.1 Naive representation

The very direct approach to build the features is that we take the input sign placement (sign positions and sign directions) as the feature and the survival rate as the label.

Let $|A|$ be the number of arcs of the graph representing the city map. We can represent a sign placement as a list of arcs where the signs are placed. The feature of an instance in this case is a vector L of $|A|$ binary values in which $L_i = 1$ means that there is a sign at arc i^{st}, while $L_i = 0$ means that there is no sign at arc i^{st}. The label now is the survival rate. For example: the figure 7.1 shows a graph representing a map of a block in the city; This graph is only for demonstration of the method, not for the experiment. The example is described as follows:

- number of vertices is 8 ($|V| = 8$)
- number of arcs is 18 ($|A| = 18$)
- there is one shelter at vertex 2 and another at vertex 4
- number of signs is 3 (K = 3)
- the first sign locates at vertex 1 pointing to vertex 0; the index of arc (1,0) is 15
- the second sign locates at vertex 7 pointing to vertex 6; the index of arc (7,6) is 16
- the third sign locates at vertex 6 pointing to vertex 3; the index of arc (6,3) is 1
- the feature vector is described as a list L of 18 elements in which $L_1 = 1$; $L_{15} = 1$; $L_{16} = 1, \forall i \in \{0, 1, 2, .., 17\} \setminus \{1, 15, 16\}$. The second row (in bold) in the table 7.1 shows the feature vector of this sign placement.
- the label is 0.4, which means that the simulation evaluates that this sign placement saves 40% of population in the evacuation

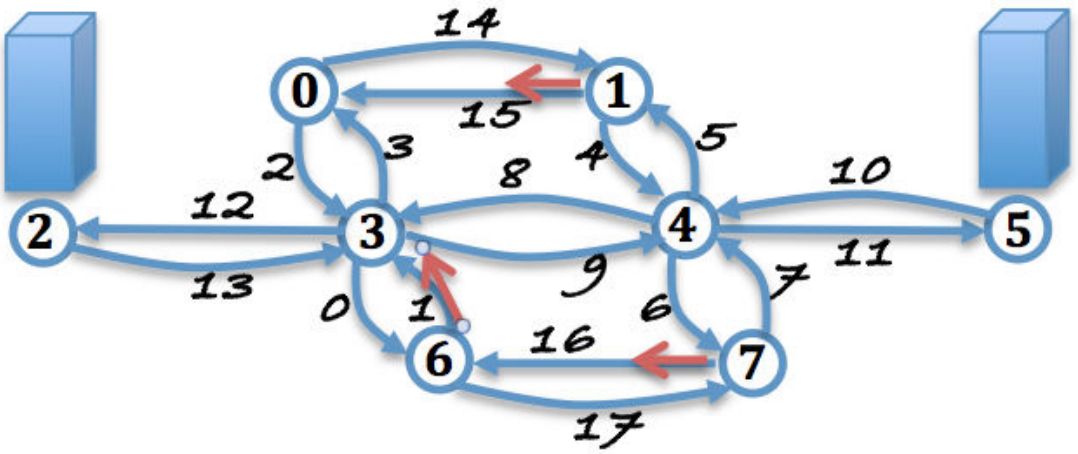


FIGURE 7.1: An example graph of a block in the city

arc 0	arc 1	.....	arc 14	arc 15	arc 16	arc 17	survival rate
0	**1**	.....	**0**	**1**	**1**	**0**	**0.4**
1	0	.....	0	1	0	1	0.6
1	1	.....	0	0	0	0	0.3

TABLE 7.1: Example of a feature vectors

arc 0	arc 1	.....	arc 15	arc 16	arc 17	**# successives**	survival
0	**1**	.....	**1**	**1**	**0**	**2**	**0.4**
0	1	.....	0	0	1	3	0.5
1	0	.....	0	1	1	1	0.3

TABLE 7.2: Add a feature to naive presentation, feature value is the number of successive signs

7.2.2 Improvement of naive representation

In the previous representation, we proposed to represent directly feature vector by sign placement. We represent a feature vector by a list of arcs where signs are placed. This representation has a flaw that each sign is independent to others, which contradicts to the fact that a group of successive signs makes the sign placement better. This is the reason why we propose an improvement of this representation.

In this improvement, we propose to add another feature into this feature vector. The value of this feature is computed by counting the successive arcs. For an example from the figure 7.1, the 1^{st} arc (6, 3) follows the 16^{th} arc (7,6), a new adding feature has value of 2 because there are 2 signs successive one another (7,6) and (6,3). The second row in the table 7.2 shows the feature vector in which we add number of successive signs.

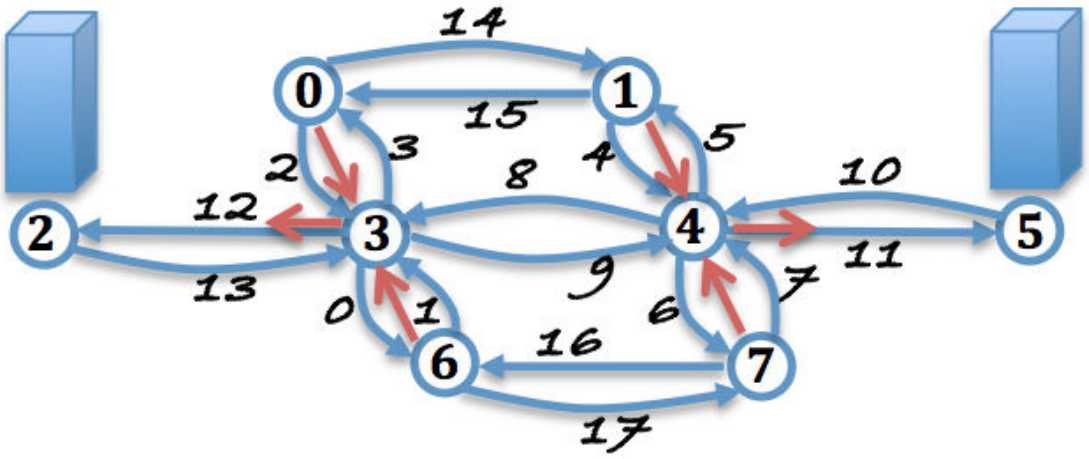


FIGURE 7.2: Example of pre-computed direction on the shortest path to the nearest shelter

vertex 0	vertex 1	vertex 2	vertex 3	.....	vertex 7	survival
0	**1**	**0**	**1**	.....	**0**	**0.7**
1	0	0	0	.....	0	0.3
1	0	0	1	.....	1	0.6

TABLE 7.3: Example of feature vector in pre-computed direction approach

7.2.3 Position with pre-computed direction representation

In the two previous representations, we represent the sign placement by th arcs where signs are placed. This approach leads to a problem that the feature has too many variables. Then, the learner needs many examples to train correctly. Here, we propose another representation using much less features.

The idea of this representation comes from the purpose of sign placement. It is true that the most important purpose of sign placement is to lead the evacuees to the nearest shelter. We propose here the idea that we pre-compute the direction of every single sign. The direction of a sign at a certain vertex in this case is the next vertex of the shortest path of this sign to the nearest shelter. The figure 7.2 shows the pre-computed direction of all the candidate positions for sign placement.

The feature vector length then is reduced from m (number of arcs) to n (number of vertices). From the example of figure 7.1, we make the pre-computation of the direction which is shown in figure 7.2. More details in figure 7.3, if we make a sign placement of 2 signs (one at vertex 1, another at vertex 3), these two signs are (1,4) and (3,2). The feature vector in this case described only the vertex indexes of the chosen sign position. The second row in the table 7.3 shows the feature vectors for this case.

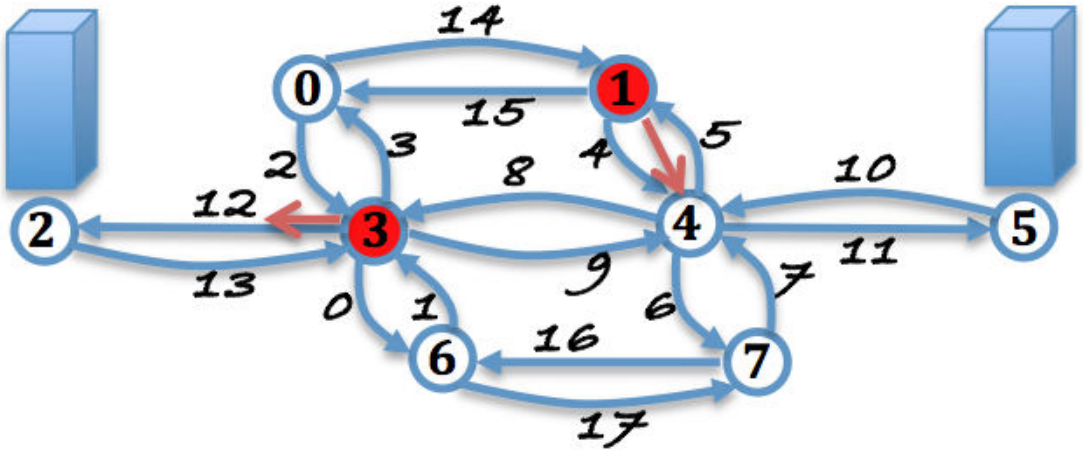


FIGURE 7.3: Example of 2 signs in pre-computed direction approach

7.2.4 Distribution representation

With regards to the nature of the evacuation, we realize that a certain sign placement changes the distribution of evacuees during the time. In fact, the evacuation is a problem of space and time. The fact that an evacuee is considered dead or alive totally depends on where and when he is at the end of the evacuation. The distribution of people at the end of the evacuation directly impacts to survival rate. Thus, our problem becomes how to build the feature describing the people distribution from the input sign placement.

At the beginning of our work, we believed that the distribution representation gave good results but it did not. However, this is still an interesting approach. Then, we decided to put the details of this approach in the appendix chapter for interest readers. In the experimentation (later sections), we mention also the result of this approach in order to compare it with other approaches.

7.3 Evaluation and comparison with Linear Regression

7.3.1 Qualitative evaluation

In order to predict the survival rate, we use linear regression. For each instance (each placement of k signs), we compare the survival rate computed from simulation (called true survival) and survival come from prediction (called predicted survival). The figure 7.4, 7.5, 7.6, 7.7 shows the result of a case of number of signs is equal to 8% number of vertices (k=8%n). In this case of experiment, we use linear regression to predict survival rate.

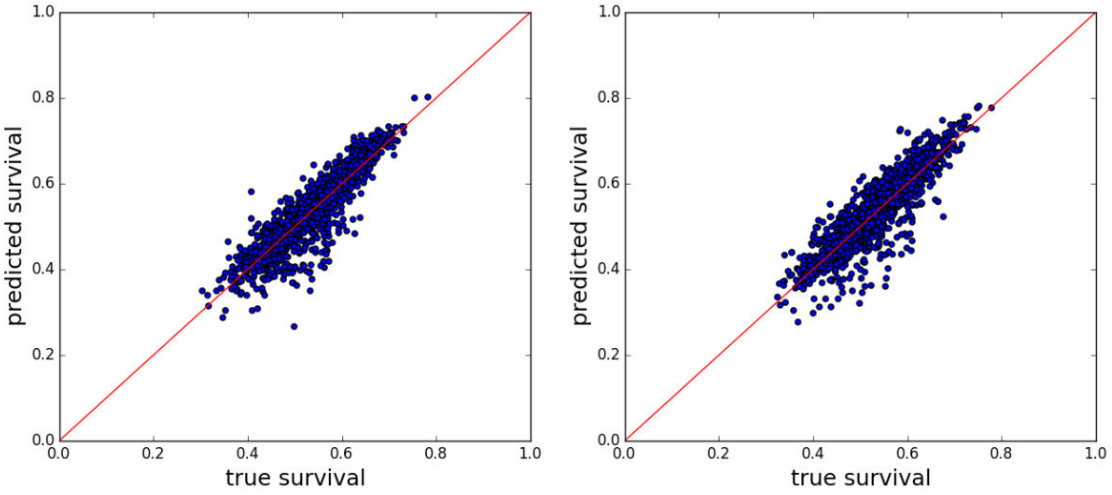


FIGURE 7.4: Result of linear prediction of naive representation

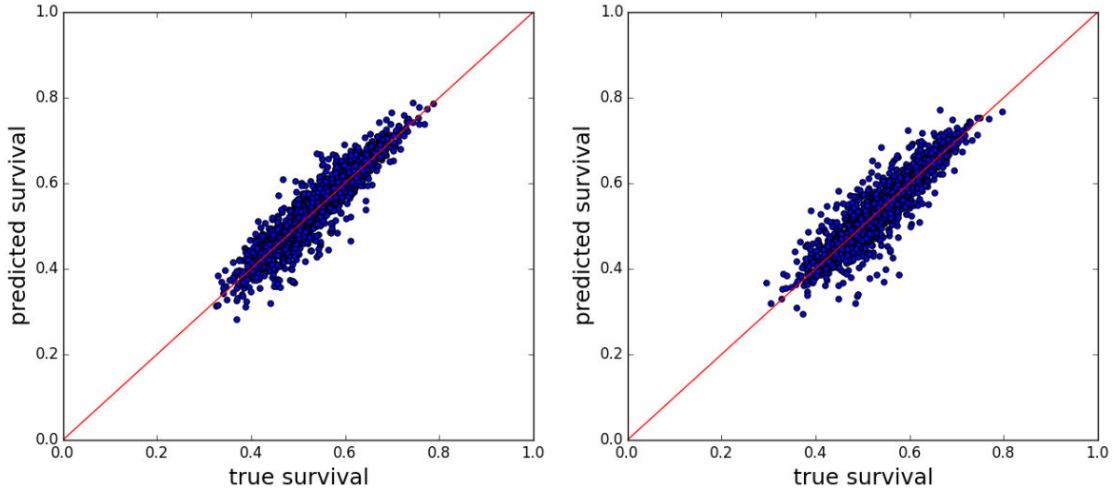


FIGURE 7.5: Result of linear prediction of naive representation adding number of successive signs

From the observation, we can see that the "Position with pre-computed direction representation" gives the promising result. However, it is difficult to compare the results from other representations. Then, it is necessary to have a quantitative evaluation.

7.3.2 Quantitative evaluation

Before making the quantitative evaluation, we must choose a measure. In this case we choose the coefficient of determination R^2 of the prediction (called score). Score is defined as (1 - u/v), where u is the regression sum of squares $\sum(yTrue - yPredict)^2$ and v is the residual sum of squares $\sum(yTrue - yTrue.mean())^2$. Best possible score is 1.0, lower values are worse.

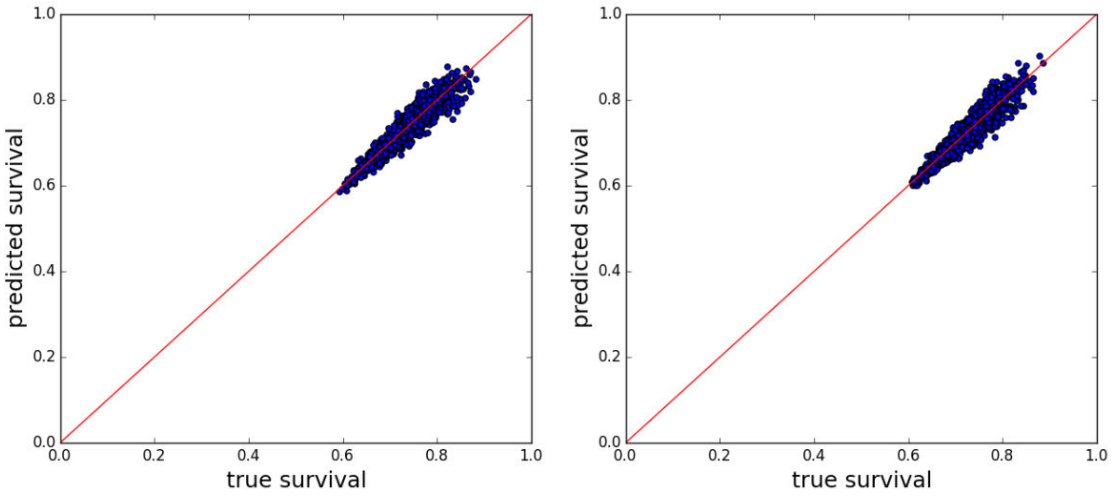


FIGURE 7.6: Result of linear prediction of precomputed direction of shortest path to nearest shelter

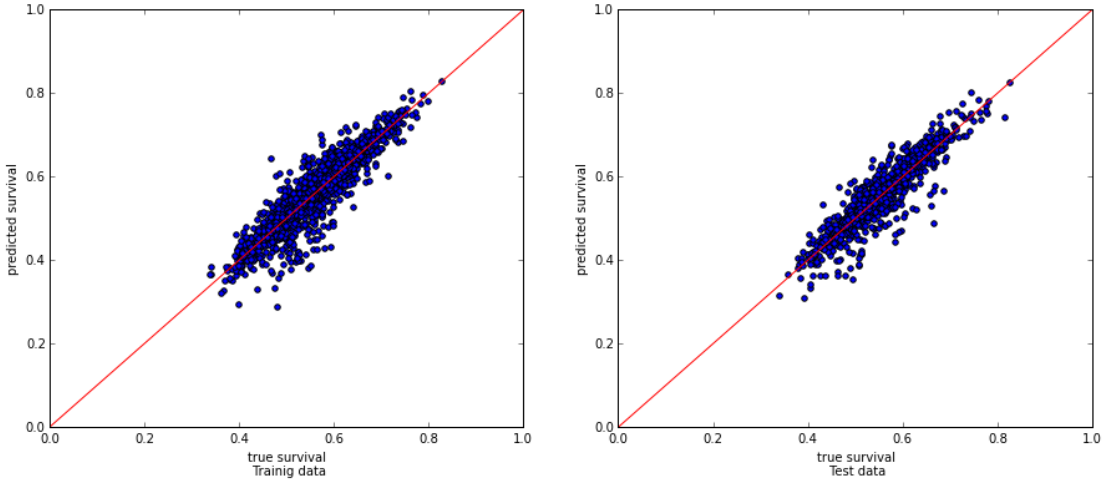


FIGURE 7.7: Result of linear prediction of distribution representation, the feature are built from average values of distribution on beacons

First, we noted that the distribution approach indeed has more than one different policy to build feature data set which might change the exactitude of the prediction. The way to choose the beacons and also the method to transform sign placement data (sign position and sign direction) into distribution feature data also give the different results. Then, we propose to begin the quantitative evaluation with the distribution representation. Once we find out the best policy to choose beacons and also the best method to build feature, we can easily make a comparison with all other representations.

As describing in the previous section, we have 4 approaches for representing the features:

- naive representation: a sign placement is a set of signs in which each sign is described by position and direction. This approach is simply call "position and direction" approach.

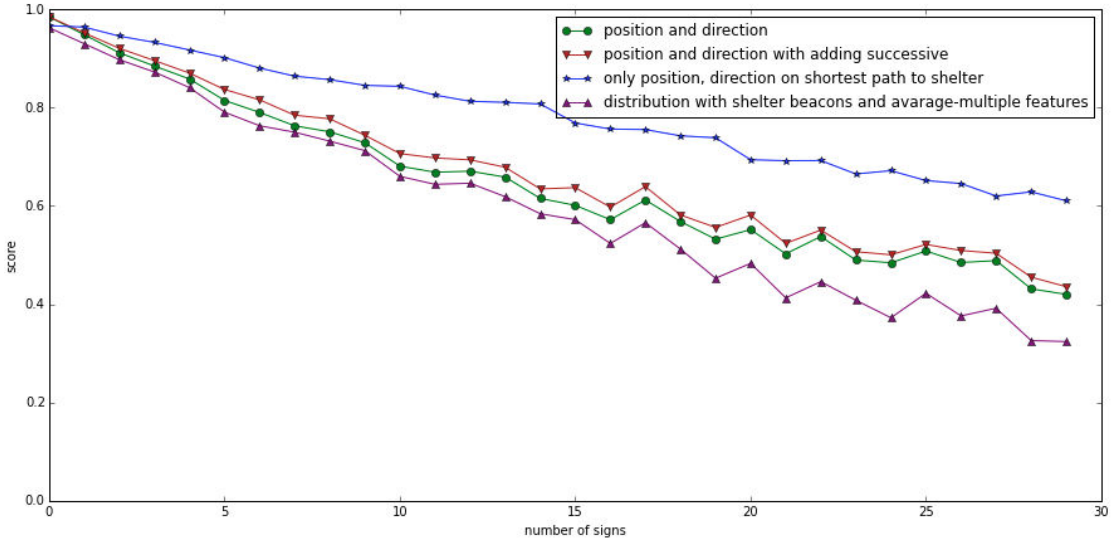


FIGURE 7.8: Compare representing approaches on test data set

- improvement of naive representation: from the naive approach, we add a new value into feature vector for each instance. This value is the number of successive signs. We can call this "position and direction, with adding successive"
- position with pre-computed direction representation: a sign placement is just a set of sign positions. The direction of each sign is pre-computed by taking the shortest path to its nearest shelter.
- distribution representation: a sign placement is transformed into feature vector by considering the density of evacuees at beacons. The beacons in this case are the shelter vertices.

In this section, we find out which is the best approach to representing the features. The figure 7.8 shows that the approach of position with pre-computed direction representation gives the best prediction. For the pre-computed direction representation, it is true that the more signs we add into the simulation, the better survival rate becomes, which is called "monotonic".

7.4 Optimizing sign placement in order to maximize survival rate using regression

The most important purpose of our work is to find out which is the best sign placement (the sign positions and sign directions such that the simulation returns the maximum survival rate). In this section, we propose how to use our prediction to find out the optimal solution to our problem.

7.4.1 Motivation

In the previous section, we show (by experimentation) that the approach of position with pre-computed direction representation is the best representation among proposed approaches. The correctness of this prediction is also acceptable. Then, we can use the linear prediction to obtain the survival rate very quickly, which allows us to run optimizing algorithm to find out the best solution. For a concrete example for this case, we can use Genetic Algorithm in which the sign placements are coded into chromosomes and the fitness of each chromosome is the survival rate evaluated by linear prediction. The optimal solution of the Genetic Algorithm is evaluated again by Agent-Based Simulation.

Our first motivation is that the Genetic Algorithm might make a blind search because the evaluational steps of this algorithm use the stochastic mutation and crossover. More specifically, at a certain generation, the algorithm has a set of evaluated solutions (the sign placements). It generates other solutions by crossover operator and mutation operator. The question here is if the new generated solutions are closer to the optimal solution than the existed ones.

The second motivation is that we have a "learner" by doing regression on existed evaluated samples. The samples here are the sign placements (and are also the solution in the aspect of the Genetic Algorithm). It is true that the "learner" can predict the survival rate, which motivates us to benefit it to do the exploration. Before going to details of our proposed algorithm, we make a short survey on the studies relating to our problem.

7.4.2 Related works

In chapter 3, we made our optimization by using Genetic Algorithm. We also improved our approach by surrogate model as well as decomposable approach. Although the result was relatively good, we decided to make a research about the optimization algorithms, hoping to find a better one.

As far as we know, Genetic Algorithm belongs to Evolutionary Algorithm. When we made a research into Evolutionary Algorithm, we found CMA ES (Covariance Matrix Adaptation Evolution Strategy) (in HANSEN [2006], HANSEN and KERN [2004], ROS and HANSEN [2008]). While the Genetic Algorithm focuses on how the population grows, the CMA ES focus on both population and distributions. However, these two algorithms are not fit our problem. In our optimization problem, we have a certain number of solutions evaluated Agent-Based Simulation, we want an algorithm who can help us to choose wisely other solutions so that they get closer and closer to the optimal one.

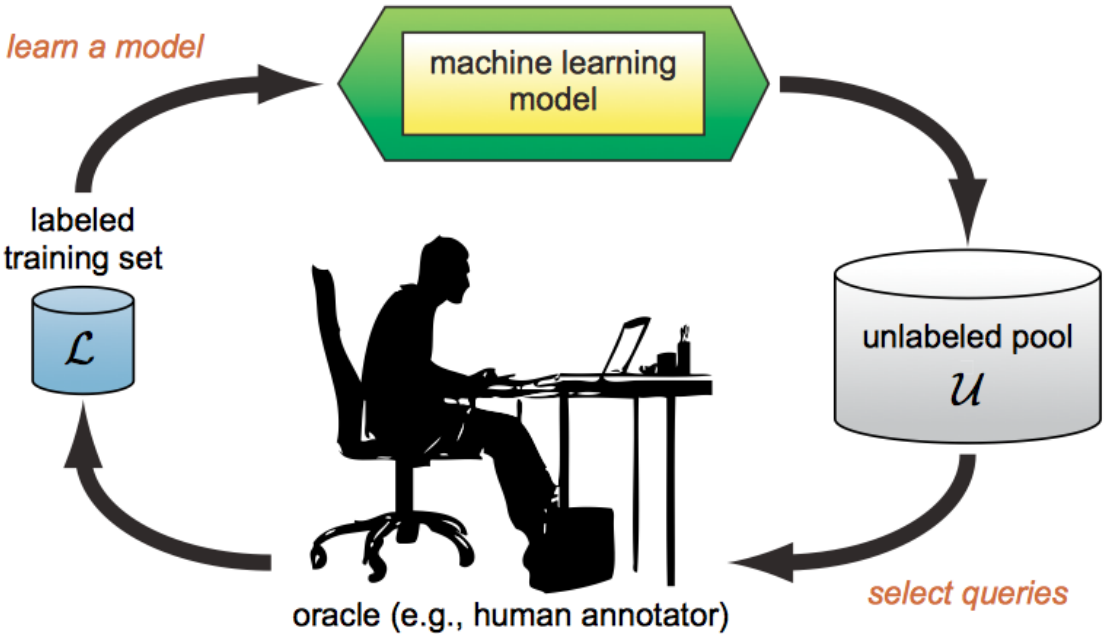


FIGURE 7.9: The pool-based active learning cycle in SETTLES [2010]

According to our survey on the documents in machine learning domain, we find that our motivations are very close to a famous approach of this domain. This approach is called "Active Learning". Moreover, our first motivation is exactly the pool-based sampling of this approach. In SETTLES [2010], the author presented "the pool-based active learning cycle", which motivates to read other studies of this approach. The figure 7.9 illustrates this cycle in which:

- The learner begins to train a small number of samples which are already labeled by the oracle.
- The oracle in the original example is the human annotator. In our case, the oracle is represented by the simulation.
- The learner learns from the labeled training set ($\mathcal{L}$)
- The learner uses its knowledge to choose (from the unlabeled pool $\mathcal{U}$) which instances to query next.
- The oracle labels the new instances.
- The new labeled instances are added to labeled set ($\mathcal{L}$).
- The cycle repeats until it reaches a termination condition.

The active learning is very useful for the problem in which the evaluation of a certain instance is very expensive and time-consuming, which is very similar to our evaluation

by Agent-Based simulation. The most important contribution of active learning is that it can select the most informative samples. Moreover, even when the learner gives a bad prediction, the correction of the oracle can benefit this bad prediction in order to make the next prediction better, which is similar to dichotomy. In LONG et al. [2010], authors have proposed a framework named Expected Loss Optimization (ELO) which is applied for ranking. The important idea of the framework is that "given a loss function, the samples minimizing the expected loss are the most informative ones". Another study illustrating the usefulness of this approach was presented in BRANKE et al. [2009]. The authors implemented a robust ordinal regression on the data resulting from the interaction with human users.

While these mentioned studies do not address to solve the optimization problem, that the way they benefit the informative samples motivates us to use it to make a well-directed exploration for our optimization problem.

7.4.3 Description of argmax algorithm

As far as we know that the result of a training using regression is a weight vector which is used to make the prediction. The idea of this algorithm is to benefit this weigh vector as the knowledge of the learner. The learner uses this weight vector to select the unlabeled samples for the next evaluation. More specifically, the unlabeled samples who are predicted (based on the current value of the weight vector) to give the maximal survival rates are selected. The selected unlabeled samples are evaluated by the simulation which plays a role as an oracle. While the learning cycle repeats, the weight vector is updated by training the labeled data.

Here, we propose another algorithm to optimize the sign placement, the argmax algorithm (algorithm 1).

In the algorithm 1, the variables are described as follows:

- K is the number of signs for a sign placement,
- I is the number of initial sign placements (and also the number of instances for the training)
- Q is the number of adding sign placements (or instances) at an iteration
- EModel is the evacuation model in order to evaluate the survival rate

Algorithm 1 Argmax algorithm

1: **procedure** ARGMAX($K, I, Q, EModel, Learner$)
2: $features \leftarrow GenerateInstances(I, K)$ ▷ generate randomly I sign placements
3: $labels \leftarrow Evaluate(features, EModel)$ ▷ Evaluate survival rate
4: $W \leftarrow InitialTrain(Learner, features, labels)$ ▷ Train by regression
5: $maxSurvivalRates \leftarrow []$
6: **while** $NotTerminationCondition$ **do**
7: $addingFeatures \leftarrow GeneratedNeighbors(W, Q)$
8: $addingLabels \leftarrow Evaluate(addingFeatures, WModel)$
9: $features \leftarrow features.append(addingFeatures)$
10: $labels \leftarrow labels.append(addingLabels)$
11: $W \leftarrow Train(Learner, features, labels)$ ▷ Train by regression
12: $maxSurvivalRates \leftarrow maxSurvivalRates.append(max(labels))$
13: **return** $maxSurvivalRates$ ▷ Return all max survival rate

- Learner is used to train the data in order to get the weight vector. The learner for each case of implementation is chosen basing on the purpose of the test case. Different learners could be: Ridge for Linear Regression, SVC for Ordinal Regression or SGDClassifier for Partial fit ordinal regression.
- W is the weight vector of the Regression "Learner"
- maxSurvivalRates is a vector which stores the max survival rates of all the iterations

For each iteration of the algorithm, the algorithm generates Q new sign placements (7^{th} line of code). The key idea of the algorithm is that the new generated sign placements are near the best sign placement (the sign placement gives the maximum survival rate). The best sign placement in this case is defined by the weight vector (W).

In this case of regression, the label (and also the survival rate) is predicted by the scalar production of a weight vector and the feature vector. In order to generate Q numbers of the sign placements near the best one, we only find the Q sets. Each set has K elements. Each element represents the index of the biggest values of w_i in W. Thus, the weight vector W plays a role as the navigator of the exploration of sign placement in order to reach the best one.

7.5 Optimizing sign placement by argmax with linear regression

From this section, we present the experiment and evaluation of the argmax using different regression methods. Here, we propose to begin experimenting the linear regression.

7.5.1 Implementation

In order to save execution time, we use the Linear Programming model of Markov chain decision to evaluate the survival rate all the cases of implementations. The scenario of the tsunami is the same as that of other Genetic Algorithm implements. The map is small. Number of agents is 1000. Number of signs is 10. Evacuation time is 900 seconds (corresponding to 15 minutes).

In order to make the comparison, we build 3 implementations:

1. Genetic Algorithm in which chromosomes are represented by sign positions, sign directions are pre-computed Direction by shortest path to the nearest shelters (called "GA positions, shortest path directions")
2. Argmax Linear Regression in which features are represented by both sign positions and sign directions. (called "LR positions and directions")
3. Argmax Linear Regression in which features are represented by only sign positions, sign directions are pre-computed Direction by shortest path to nearest shelters (called "LR positions, shortest path directions")

7.5.2 Qualitative evaluation

It is curious and also necessary that we want to see how the result of the Argmax optimization looks like. In this evaluation, we run the Argmax on Linear Regression with the naive representation. In order to clarify the analysis, we note also the Argmax survival rate, the Average survival rate along with the best survival rate.

The figure 7.10 shows the result of the argmax. Focusing on the curves representing the best survival rate and the average survival rate, we find that these curves are stably increasing. The reason for this result is that the adding sign placements get better and better. The most unstable curve is the argmax survival rate. The argmax survival rate is computed directly from the Weigh vector. While this curve is highly unstable at the beginning of the algorithm, it is gradually converging, which confirms correctness of the algorithm.

7.5.3 Quantitative evaluation of argmax of linear regression

The purpose of this evaluation is to find out if this optimization method is better than the previously proposed ones, which means that we compare here the Argmax Algorithm

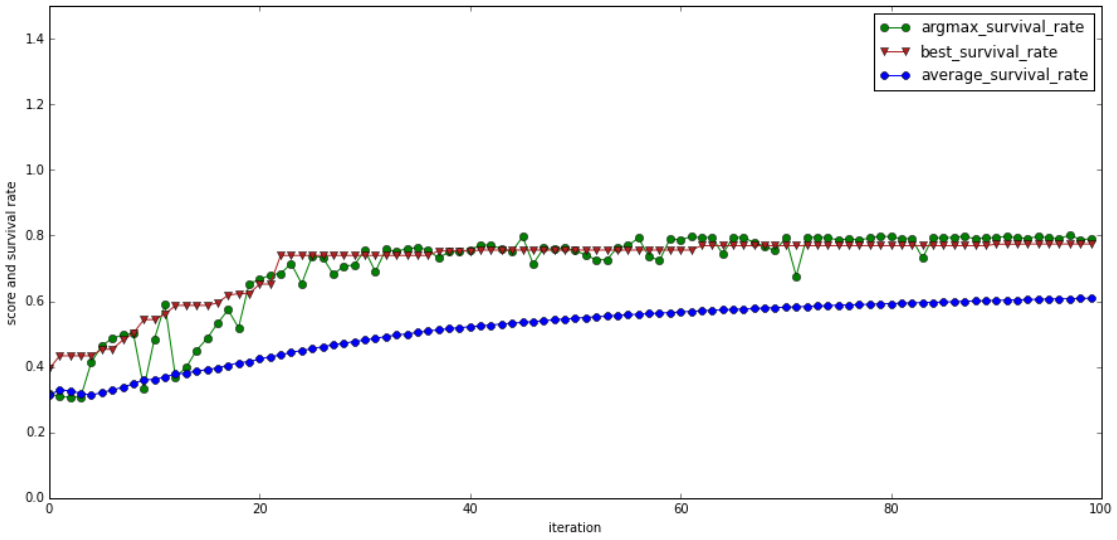


FIGURE 7.10: Argmax algorithm with linear regression using naive representation

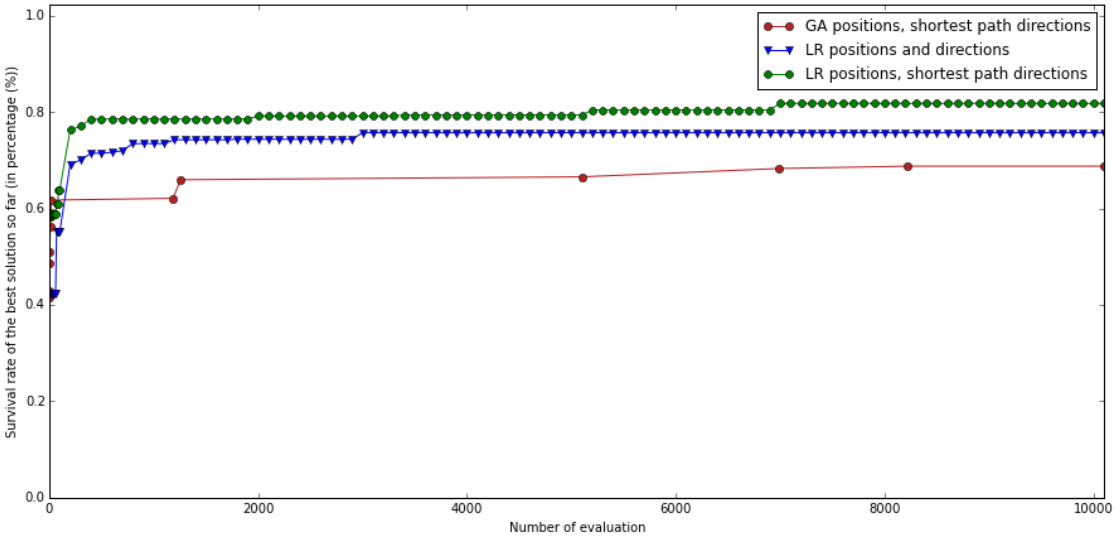


FIGURE 7.11: Compare Genetic Algorithm with Argmax of Linear Regression

with the Genetic Algorithm. In the previous chapters, we propose different implementations of Genetic Algorithm for optimization of sign placement. In this evaluation, we choose the best Genetic Algorithm implementation (called "GA positions, shortest path directions") to be compared with the Argmax.

In figure 7.11, we easily see that the Argmax Algorithm prevails the Genetic Algorithm even when the Argmax uses the naive representation of the features. In this case, the Argmax in which the features are represented by the sign positions and sign directions are computed by shortest path (called "LR positions, shortest path directions") is the best among three implementations. The reason for this performance of the Argmax is the way that the Argmax generates instances for the next generation. The next instances

here are around the best solution so far which are defined by the Weight vector of the Learner.

7.6 Optimizing sign placement by argmax with ordinal regression

In this section, we present the ordinal regression and how to use it to do the optimization.

7.6.1 Motivation and related works

Firstly, we re-emphasize that our purpose is to find the sign placement which results the maximal survival rate. This purpose can be implied that we do not need to directly predict the exact survival rate. Rather, we only need to predict which sign placement gives greater value of survival rate than others. All these ideas motivate us to study ordinal regression.

On a short survey on the related works, we found that the study in INGIMUNDARDOTTIR and RUNARSSON [2011] is the most relevant to our work. In this document, the authors proposed to a model (called surrogate model) which indirectly predicts the ranks of samples by using ordinal regression. They also introduce a new validation/updating policy for this surrogate model.

In our problem, we first re-used the idea of the ordinal regression usage and then proposed the modification of argmax algorithm in order to fit the ordinal regression. For our case, the 20% best instances are considered good. The other 80% are considered bad. The table 7.4 presents 10 instances which are sorted by descending survival rate. The 2 top samples (ex0 and ex1) are considered good. The below samples are considered bad. The good class has the label value 1, while the bad one denotes -1 as the label.

7.6.2 Argmax algorithm with ordinal regression

This argmax algorithm with ordinal regression is the first version of argmax algorithm with only one modification. The modification is that we add only one phase which separated the data into good class and bad class. In the algorithm 2, this modification is presented at line 4,5 and line 12, 13. Line 4 and line 12 execute the separation of data into good class and bad class. Line 5 and line 13 train the newly obtain data.

	arc 0	.....	arc 14	arc 15	arc 16	arc 17	survival rate
ex0	**0**	.....	**1**	**0**	**1**	**0**	**0.7**
ex1	**1**	.....	**1**	**1**	**0**	**1**	**0.6**
ex2	0	.....	1	1	1	0	0.5
ex3	1	.....	0	0	1	1	0.5
ex4	1	.....	0	1	0	0	0.4
ex5	1	.....	1	1	1	1	0.4
ex6	1	.....	1	0	1	1	0.3
ex7	0	.....	0	1	0	1	0.2
ex8	1	.....	0	1	1	0	0.2
ex9	0	.....	1	0	1	1	0.1

TABLE 7.4: Example of a feature vectors illustrating good class and bad class

Algorithm 2 Argmax algorithm with ordinal regression

procedure ARGMAX($K, I, Q, EModel, Learner$)
  $features \leftarrow GenerateInstances(I, K)$ ▷ generate randomly I sign placements
  $labels \leftarrow Evaluate(features, EModel)$ ▷ Evaluate survival rate
  $[pFeatures, pLabels] \leftarrow Separate(features, labels)$ ▷ Separate good/bad
  $W \leftarrow InitialTrain(Learner, pFeatures, pLabels)$ ▷ Train by regression
  $maxSurvivalRates \leftarrow []$
  while $NotTerminationCondition$ **do**
    $addingFeatures \leftarrow GeneratedNeighbors(W, Q)$
    $addingLabels \leftarrow Evaluate(addingFeatures, WModel)$
    $features \leftarrow features.append(addingFeatures)$
    $labels \leftarrow labels.append(addingLabels)$
    $[pFeatures, pLabels] \leftarrow Separate(features, labels)$ ▷ Separate good/bad
    $W \leftarrow Train(Learner, pFeatures, pLabels)$ ▷ Train by regression
    $maxSurvivalRates \leftarrow maxSurvivalRates.append(max(labels))$
  return $maxSurvivalRates$ ▷ Return all max survival rate

7.6.3 Implementation and evaluation of argmax of ordinal regression

In this case, we compare the Genetic Algorithm with the Argmax with Linear Regression and with the Argmax with ordinal regression. We then build 5 implementations:

1. Genetic Algorithm in which chromosomes are represented by sign positions, sign directions are pre-computed Direction by shortest path to nearest shelters (called "GA positions, shortest path directions")

2. Argmax Linear Regression in which features are represented by both sign positions and sign directions. (called "LR positions and directions")

3. Argmax Linear Regression in which features are represented by only sign positions, sign directions are pre-computed Direction by shortest path to nearest shelters (called "LR positions, shortest path directions")

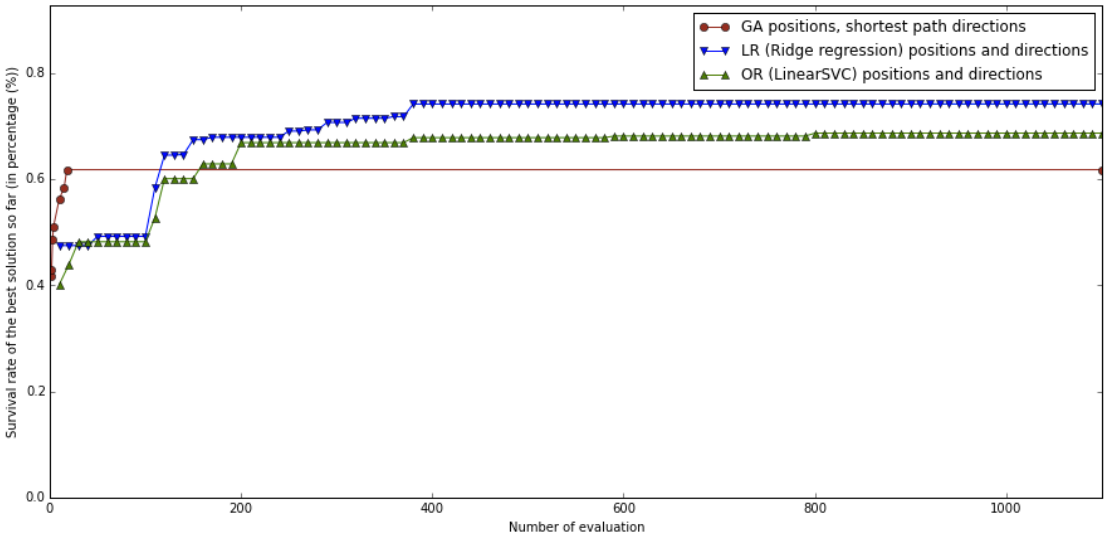


FIGURE 7.12: Compare Genetic Algorithm with Argmax of Ordinal Regression with naive representation

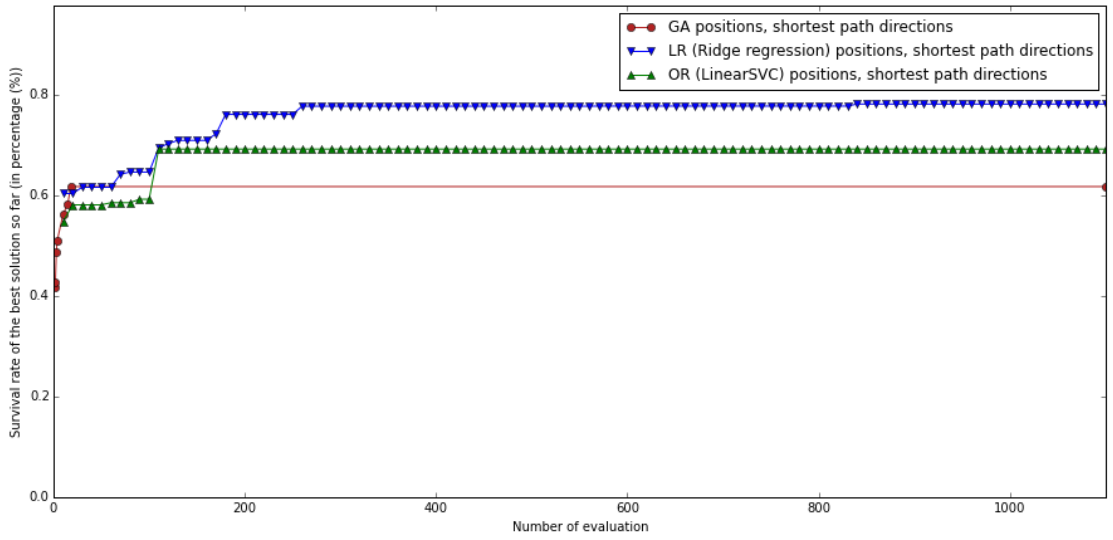


FIGURE 7.13: Compare Genetic Algorithm with Argmax of Ordinal Regression with pre-computed direction representation

4. Argmax Ordinal Regression in which features are represented by both sign positions and sign directions. (called "OR positions and directions")

5. Argmax Ordinal Regression in which features are represented by only sign positions, sign directions are pre-computed Direction by shortest path to nearest shelters (called "OR positions, shortest path directions")

In the evaluation, we test the performance of three optimization methods with both naive representation and precomputed direction representation. The figure 7.12 shows that both regression methods are better than the Genetic Algorithm in naive representation. For the precomputed direction representation, the figure 7.13 shows the same result

that the regression methods prevails. Between these two regression methods, the Linear Regression is better.

7.7 Optimizing sign placement by argmax with ordinal regression with partial fit

7.7.1 Time-consuming training phase

While the number of samples does not increase execution time of Linear Regression, it does with ordinal one. In order to overcome this issue, we propose to use partial fit to train the data in the repeating active learning cycle. Once we use the partial fit, we do not need to accumulate the sample. The newly generated samples replace the old ones. This idea means that the number of samples for partial fit rests the same during the repeating cycle. This is in sharp contrast to the Linear regression that the number of samples increases during the repeating cycle.

7.7.2 Argmax algorithm with ordinal regression with partial fit

Basically, this version of algorithm (algorithm 3) is similar to algorithm 2 except the changes in line 10, 11, 13. In line 10 and line 11, the features data and the labels data are replaced by the new generated data. In line 13, the PartialFit method is invoked instead of the Train method.

Algorithm 3 Argmax algorithm with ordinal regression with partial fit

1: **procedure** ARGMAX($K, I, Q, EModel, Learner$)
2: $features \leftarrow GenerateInstances(I, K)$ ▷ generate randomly I sign placements
3: $labels \leftarrow Evaluate(features, EModel)$ ▷ Evaluate survival rate
4: $[pFeatures, pLabels] \leftarrow Separate(features, labels)$ ▷ Separate good/bad
5: $W \leftarrow InitialTrain(Learner, pFeatures, pLabels)$ ▷ Train by regression
6: $maxSurvivalRates \leftarrow []$
7: **while** $NotTerminationCondition$ **do**
8: $addingFeatures \leftarrow GeneratedNeighbors(W, Q)$
9: $addingLabels \leftarrow Evaluate(addingFeatures, WModel)$
10: $features \leftarrow addingFeatures$
11: $labels \leftarrow addingLabels$
12: $[pFeatures, pLabels] \leftarrow Separate(features, labels)$ ▷ Separate good/bad
13: $W \leftarrow PartialFit(Learner, pFeatures, pLabels)$ ▷ Train by regression
14: $maxSurvivalRates \leftarrow maxSurvivalRates.append(max(labels))$
15: **return** $maxSurvivalRates$ ▷ Return all max survival rate

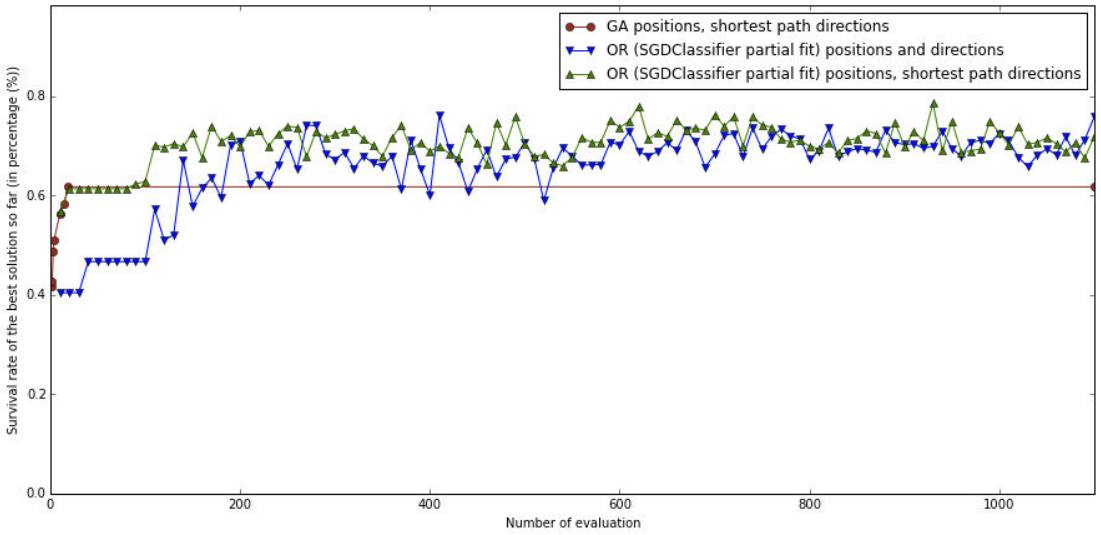


FIGURE 7.14: Compare Genetic Algorithm with Argmax of Ordinal Regression with partial fit

7.7.3 Quantitative evaluation of argmax of ordinal regression with partial fit

In this case, we then build 3 implementations:

1. Genetic Algorithm in which chromosomes are represented by sign positions, sign directions are pre-computed Direction by shortest path to nearest shelters (called "GA positions, shortest path directions")

2. Argmax Ordinal Regression with partial fit in which features are represented by both sign positions and sign directions. (called "OR partial fit positions and directions")

3. Argmax Ordinal Regression with partial fit in which features are represented by only sign positions, sign directions are pre-computed Direction by shortest path to nearest shelters (called "OR partial fit positions, shortest path directions")

The result from figure 7.14 indicates that the regression once again beats the Genetic Algorithm. On the observation, we can notice that the curves representing regression methods are zigzag. The amplification of the zigzag is large in the beginning but it reduces gradually along with the increasing of the survival rate.

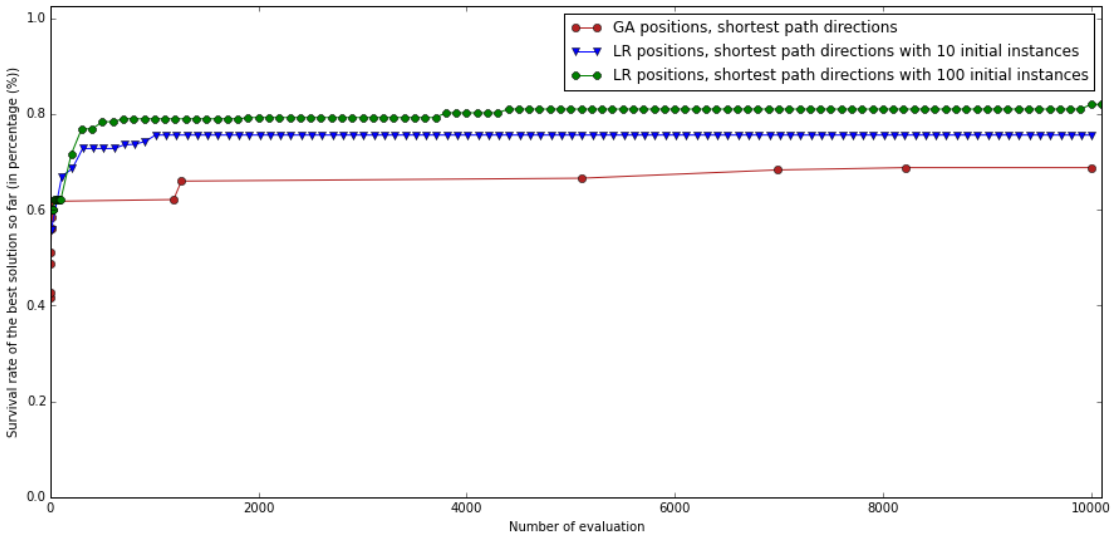


FIGURE 7.15: Compare Argmax of Linear Regression of shortest path representation with different number of initial instances

7.8 Impact of initial number of samples on performance of Argmax

Even though the Argmax shows its better results to the Genetic Algorithm, it has its own issue. On observation of the results, we find that the performance of the Argmax depends on the number of initial instances which are used to train to build the first Weigh vector. This means that the more initial instances make the better result of the Argmax. The figure 7.15 and the figure 7.16 show the result of the Argmax with the different numbers of initial instances. The 2 figures show that (in both cases of representations) 100 initial instances make the result better than 10 initial instances do. Let us focus on the figure 7.16 which shows the result of the naive representation: while 100 initial instances make the result better than the Genetic Algorithm, 10 initial instances make it worse.

As we know that the Weight vector is the most important factor of the Argmax. It can make the optimization phase better and also make it worse, which totally depends on the number of initial instances. The only reason for this phenomenon is that the first learning phase does not return a good weigh vector. In fact, if the number of samples is less than the number of variables in the feature vector, weigh vector is not general enough, which leads to the bad result.

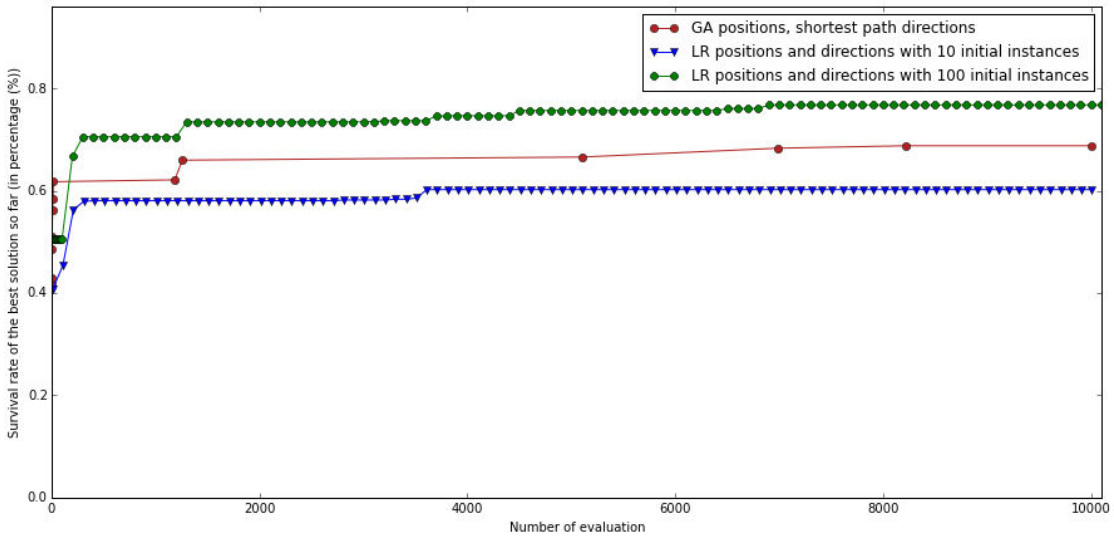


FIGURE 7.16: Compare Argmax of Linear Regression of (Positions Directions) representation with different number of initial instances

7.9 Conclusion

In this chapter, we propose another aspect of evaluation of survival rate. In this aspect, we predict the survival rate by Regression. From the regression, we also propose another optimization algorithm, the Argmax. According to the results of the experimentation, we conclude the Argmax can make the faster exploration than the Genetic Algorithm does.

In the next chapter, we propose another guiding solution for evacuation, which save significantly the evacuees.

Chapter 8

Local Evacuation Map

8.1 Introduction of the problem

In the previous chapters, we present more than one solution to optimization of sign placement in order to help people in evacuation. In this chapter, we propose another solution which is expected to significantly improve the survival rate. The solution here is called local evacuation maps.

A local evacuation map is a panel placed in the city usually near the junction (or crossroad). We call such map local because it shows only the surroundings including: the nearby safe places (called shelters) in where people can stay when the tsunami arrives, the shortest path from the current location (where the panel of the map is placed) to the nearest shelter. In figure 8.1, the map shows: the current location of evacuees, the nearest shelter and the highlighted the shortest path to nearest shelter. Once this kind of map is perceived, the evacuees follow the shortest evacuation path to the shelter.

We suppose, we have a limited budget which can be produced L maps. Our problem is where to place them in city (on which junctions or crossroads) in order to maximize number of survivors in case of a typical tsunami scenario. Of course, people crossing this map are expected to follow the suggested path to safe places. In this chapter, we first present two approaches to evaluate the survival rate of the evacuation in which Local Evacuation Maps are integrated: one uses the agent-based simulation and other uses the linear programming formulation. Then, we use this evaluation as fitness function for genetic algorithm on candidate locations of the maps, which is similar to the optimization method in Chapter 3.

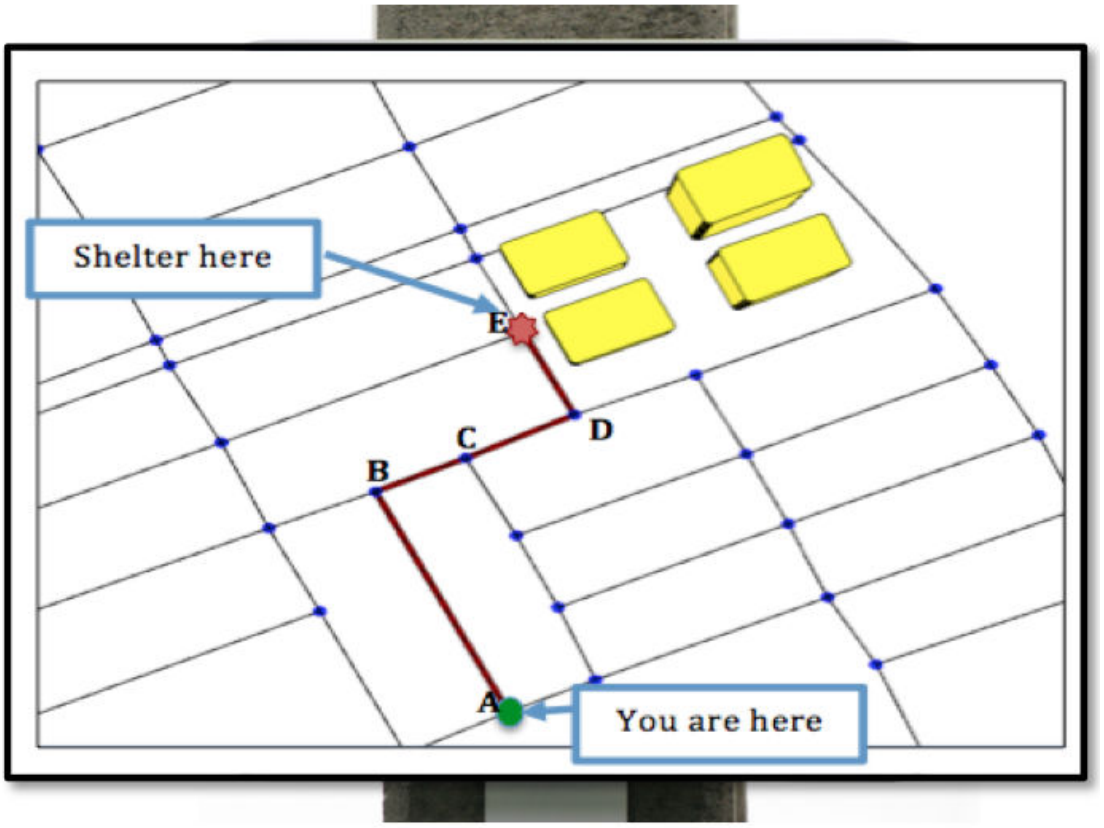


FIGURE 8.1: A proposed local evacuation map

8.2 Survey on evacuation map

Before going to the solution, we make a short survey on how people use evacuation map. We all agree that the evacuation map provides the most important information for people in evacuation. Regarding the education, evacuation map helps people to know in advance where to go when a disaster happens. KUROWSKI et al. [2011], KATADA and KANAI [2008] showed important role of evacuation map in education. It would be useful if all people are aware of the disasters and of safe places. They become active to evacuate in case the disaster happens.

In fact, the evacuation map received intensive studies in the recent years. LIU et al. [2014] built an simulation of emergency evacuation in which the evacuation map is the useful information for evacuees. KASAHARA et al. [2014] proposed "A Tourism Information Service for Safety during School Trips" in order to guild the pupils in case of tsunami or other disasters. In IMAMURA et al. [2012], local government of Padang City, Indonesia introduced evacuation map of the city in 2005 (after the tsunami disaster in 2004 which caused thousands of casualties). Although it is a late response but it is necessary to prepare for the future. SILVA et al. [2011] proposed to link open data in

order to make the preparation plan for tsunami. Among the data, the authors indicated that the evacuation map is the most important information for preparation.

While the evacuation map is proved to be important, the way to evaluate it is still unclear. The direct way to evaluate the evacuation map is to ask the citizen. DALL'OSSO and DOMILEY-HOWES [2010] proposed to make a survey on evacuation map. The authors stated that "Results of the survey indicate that residents think the maps are useful and understandable, and include insights that should be considered by local government planners and emergency risk management specialists during the development of official evacuation maps (and plans) in the future".

In our situation, we propose the local evacuation map which is considered a small part of the evacuation map. A list of local evacuation maps can play the role of the evacuation map in general. Our purpose here is to evaluate the survival rate of this list of evacuation maps.

8.3 Formalization of the optimization problem

First, the most important is how to evaluate a solution using local evacuation maps. A solution of evacuation maps is simply a set containing L local evacuation maps, which is similar to the sign placement containing a set of K guiding signs. Then, the way to evaluate a solution of local evacuation maps is nearly the same of sign placement. We have at least 2 methods to do this.

8.3.1 Fitness evaluation by Agent-Based Simulation

The first and also simple method is the Agent-Based Simulation. We build an Agent-Based simulation in which: the environment is modeled as a graph exactly like the model presented in Chapter 2. The outcome of this model is also the survival rate. The only two differences of the model evaluating signs and the model evaluating local evacuation maps are the agents' behaviors:

1. Input parameter: The input parameter of this model is the locations where the local evacuation maps are placed. More concretely, a solution of local evacuation maps contains a set of vertices. A specific local evacuation map placed at a vertex in this set.

2. Agents' behaviors: The agents representing pedestrians in this model are to carry out 2 actions: moving along the edge of the graph, choosing the next target when

it arrives a crossroad. If a local evacuation map is perceived, the agent follows the shortest path to reach the shelter, otherwise, it turns randomly.

The Agent-Based modeling provides an easy way to extend the evaluation. In our case, if we want to evaluate a solution in which we use both guiding signs and local evacuation maps, we only need to add one more behavior that: if agents perceive a guiding sign, they follow the direction of the sign. However, the Agent-Based simulation needs much time, for it has to evaluate the behaviors of every single agent while running the simulation.

8.3.2 Fitness evaluation by Linear Programming Formulation of Casualties Evaluation of Pedestrian Evacuation

As we mention in the chapter 4, we do have a linear programming method that evaluates very fast the survival rate of the pedestrian evacuation. In this case, we can reuse this Linear Programming model to evaluate the survival rate by re-modeling the sign placement into local evacuation map placement.

8.3.2.1 Modeling the placement of local evacuation map using shortcut

The first aspect to the shortest path is that we can consider it as a shortcut. Let vertex i represent the location where a local evacuation map is placed, and vertex j represent the nearest shelter. In the figure 8.2), vertex i and vertex j respectively represent the crossroad A and the crossroad E. We can suppose that there is a shortcut from vertex i directly to vertex j. The length of the shortcut is equal to the length of the shortest path from vertex i to vertex j. The probability for agents situating at vertex i to choose this shortcut is 1.0%. The probabilities for other routes are all equal to zero. The modifications of the quadruplet (G, X, μ, P) are described:

1. Add edge (i,j) to A
$$A = A \cup \{(i, j)\} \text{ (1)}$$
2. Set weight of edge (i,j) at weight of the shortest path which is the sum of all belonging edges
$$c_{ij} = \sum_{(h,k) \in ShortestPath(i,j)} c_{hk} \text{ (2)}$$
3. Set the values of the transition probabilities P so that agent always moves to vertex j
$$p_{ij} = 1 \text{ (3)}$$

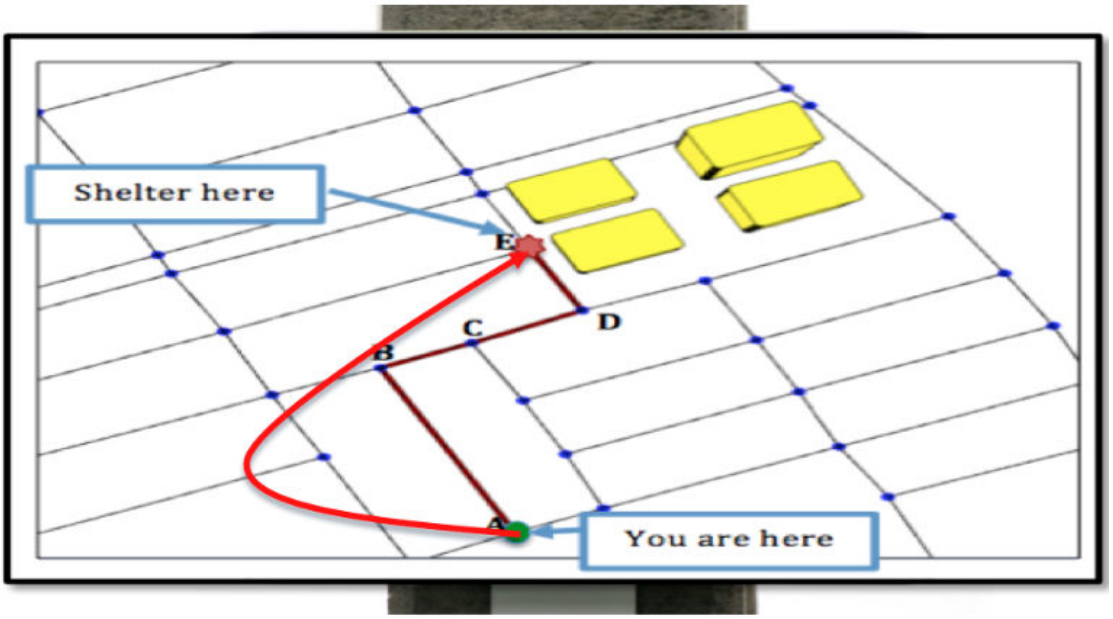


FIGURE 8.2: Local Evacuation Map represented by a shortcut

$$p_{ik} = 0, \forall k \neq j \ (4)$$

8.3.2.2 Modeling the placement of local evacuation map adding signs

Another aspect to the shortest path is that we can consider it as a set of successive signs. The figure 8.3 shows an example of successive signs. In this case, we place one local evacuation map situates at A. The shortest path to the nearest shelter is (A, B, C, D, E). Then, the adding signs are (A to B, B to C, C to D, D to E).

For each local evacuation map, we add these successive signs into the sign placement. This sign placement is added to the Linear Programming model to evaluate survival rate.

8.4 Implementation and evaluation

In this case, we compare percentage of survivors between guiding sign optimization and local evacuation map optimization. We made 2 implementations: The first one was the optimization sign placement and the second was the optimization of local evacuation map. Since the linear programming formulation and the Agent-Based simulation produced the same result, in order to save time, we chose the linear programming formulation to evaluate fitness in both implementations. We also applied the same genetic algorithm (same kind of evolutionary algorithm with the same coefficient of mutation

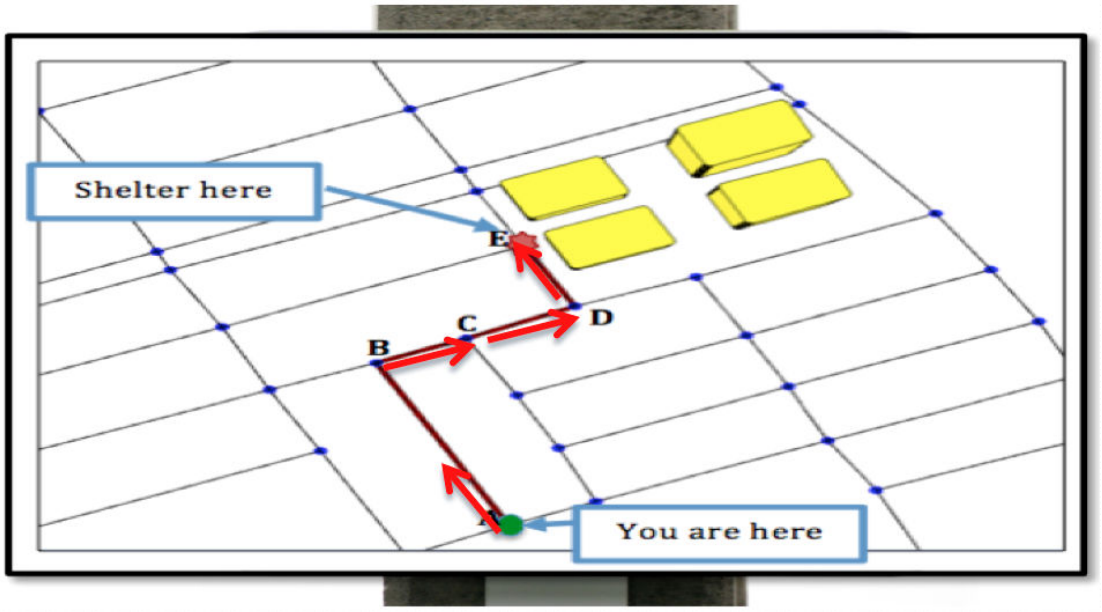


FIGURE 8.3: Local Evacuation Map represented by successive signs

and crossover) on both implementations. The genetic algorithm of both implementations was run many times on the same computer.

8.4.1 Compare guiding signs with local evacuation maps

In this case of test, we evaluate if local evacuation maps save more people than guiding signs. We run both implementations in which the number of signs is equal to the number of local evacuation maps. We repeat running these two implementation with increasing number of signs (equal to that of local evacuation maps).

The figure 8.4 shows the result of both implementation with different number of signs (or maps). The X-axis in this case represents the percentage of signs (or percentage of local evacuation maps) over the number of vertices. The Y-axis always represents the survival rate. The result indicates that local evacuation map optimization prevails guiding sign one. The reason is quite clear: a guiding sign just represents a turn at a certain crossroad and a local evacuation map provides the shortest path including a lot of crossroads which represent a lot of guiding signs.

8.4.2 Compare the best case of both approaches

In this case we analyze the best case of each implementation. This is the case when the number of signs is 16% of the number of vertices. For each implementation, the curve

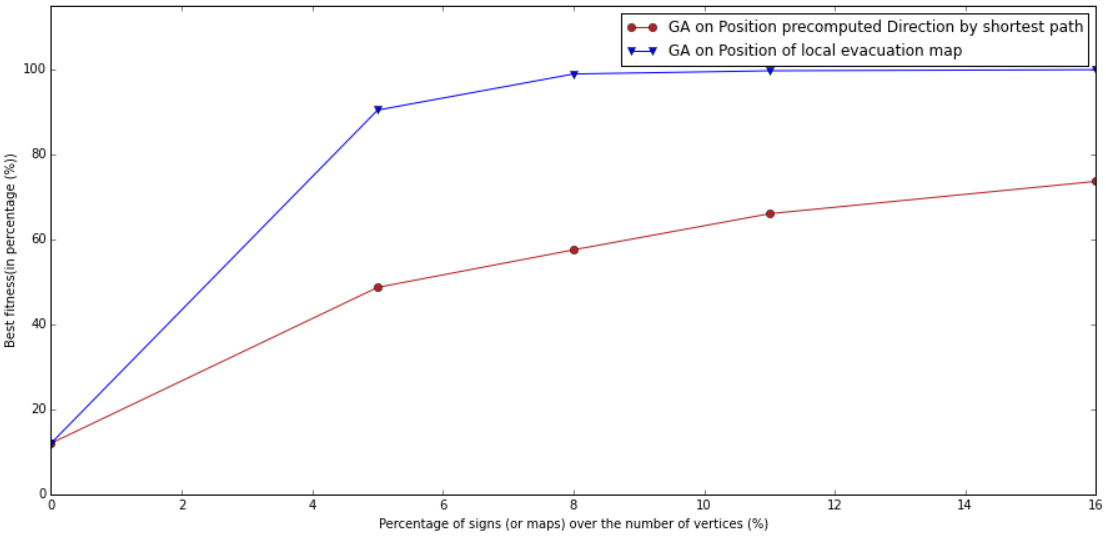


FIGURE 8.4: Comparison between guiding sign optimization and local evacuation map optimization

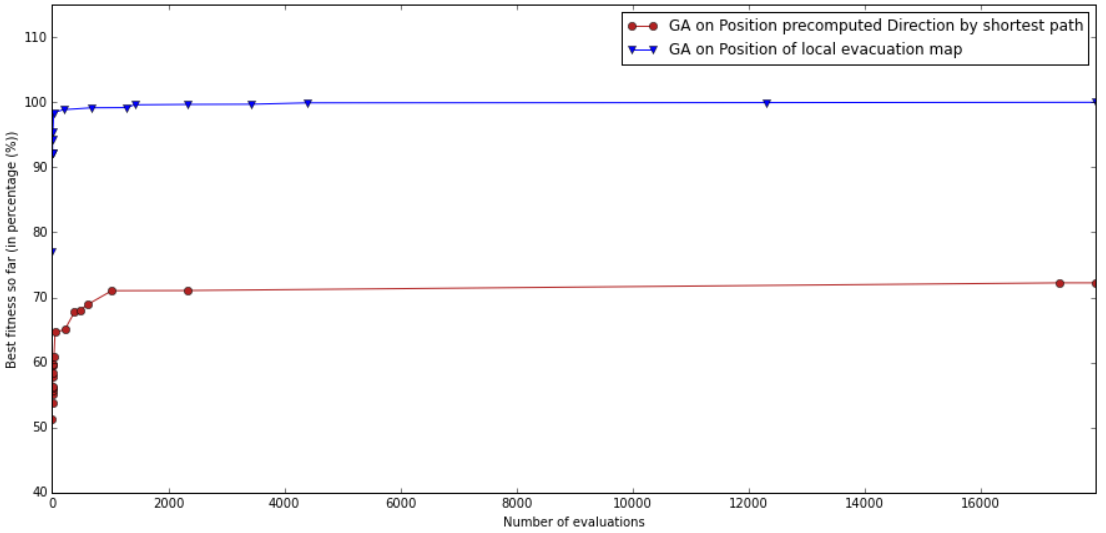


FIGURE 8.5: Comparison between guiding sign optimization and local evacuation map optimization

represents the augmentation of the survival rate over number of generations of Genetic Algorithm.

The figure 8.5 shows that the local evacuation maps is much better than the guiding signs.

8.4.3 Discussion on validity of a local evacuation map

Despite of the better performance of the local evacuation maps over the signs, there is still an issue of validity of local evacuation map. The question here is if every the

local evacuation map is valid. In fact, a person can not memorize a long path including a lot of turns along with plenty of crossroads. Then, a local evacuation map must be local, which means that it must be near its nearest shelter. It seems to us that a path should be better if it has less than five turns. Therefore, the local evacuations maps whose the shortest path has more than 5 vertices is considered invalid in our case (also in our implementations). The validity of the local evacuation opens a perspective which is described in the conclusion.

8.5 Conclusion

In this chapter, we propose another guiding information in order to help people in evacuation. This is the local evacuation map. In fact, the local evacuation map can do a job of many successive signs, which improves the survival rate.

An perspective for this approach is the combination of the signs and local evacuation maps. As we know that, a evacuation map is more expensive than a sign. In crowded areas near the shelters, it is good to place local evacuation maps. But in the less crowded areas, the signs are the good choice. The problem here is how to optimize all of them.

Chapter 9

Conclusion

9.1 Different aspects and different ideas of solutions

First, we present one important difficulty of evacuation in case of tsunami. This difficulty is that there is a part of people in the population who do not know to where to evacuate. In order to overcome this difficulty we propose to use guiding sign system which opens the problem to solve of this document. The problem is how to place these signs in order to have maximum survivors.

Regarding the aspect of complexity, this optimization problem is infeasible. However, we propose different aspects which separates this problem into tractable sub-problems, which make the whole problem becomes tractable as well. This is the main contribution of this document.

9.1.1 Aspect of black-box objective function in optimization algorithm

We first propose to use an Agent-Based simulation in order to evaluate the sign placement. Then, we propose to use Genetic Algorithm to do the optimization of sign placement. In this case, the Agent-Based simulation is similar to the black-box which plays a role as fitness function.

9.1.2 Aspect of surrogate to accelerate black-box objective function

Since the Agent-Based simulation is time-consuming approach. We propose to replace it by a surrogate model which is much faster than the Agent-Based one. This surrogate

model is, in fact, the Linear Programming model which simulates the decisions of pedestrians as a Markov decision process. Thanks to this extremely fast model, the Genetic Algorithm become tractable with respect to execution time.

9.1.3 Aspect of decomposable problem

From the fact that a guiding sign has two parts (a sign position and a sign direction), we propose to decompose this optimization problem into two sub-problems. One problem does the optimization of sign positions. Other computes the directions from obtained positions. This approach considerably accelerates the optimization phase.

9.1.4 Aspect of machine learning in prediction of survival rate

As the Genetic Algorithm does the stochastic searching which slows down the optimization phase, we propose another aspect which makes the searching phase more intelligent. In this proposition, we first propose to predict the survival rate by regression. Then, we propose an argmax algorithm to do the optimization. This algorithm is inspired from the active learning approach which benefits the informative samples (the sign placements in this case) to do searching. The experiments show that this approach significantly improves the performance of the optimization phase.

9.2 Works in progress

9.2.1 Graphic interface for user

We aim to bring this study into application under the form of a software. This software is expected to be used by the local administrators who plan to deal with the disaster. This software provides functions which allows user to choose the scenario, define parameters. The system runs and returns the results. The user makes decision to deploy the sign by consulting the results from this system. The illustrations of this software are in the appendix.

9.2.2 Parallel evaluation

Since the evaluation takes much time to run, we propose to run it on multiple machines. We did receive a relative success by benefitting the master/slave facility of ECJ. The

master part plays a role of the collectors who receives the fitness from multiple slaves. The slaves in this case do the evaluation which return the fitness to the master.

9.3 Future works

9.3.1 Impact of people distribution on disaster arrival moment

9.3.1.1 Analysis people distribution at different period of a day

As we know that the performance of the sign placement totally depends on the people distributions. In the real world, the distribution of population depends on the time. In daylight, most people are at work. In the midnight, most of them are at home. In cold winter, there are few people near the beach. In hot summer, the beach is covered by crowed people. Then, the distribution of people in the real world is much different to uniform distribution. In order to bring the solution to application, the distribution of people depending on certain period of time must be taken into account.

9.3.1.2 Proposition of balancing distribution

The idea for this problem is that we can balance the distribution of people. First, we propose to use reliable population data provided by local government. Then, we balance the people distribution based on proportion of time. We present here an example of a certain area where the local government makes the survey on the population. The survey reports that in the daylight (during 16 hours), there are (in average) 110 people. In the night (during 8 hours), there are only 10 people there. Then the distribution of population of this area is equal to $(110 \times 16 + 10 \times 8)/24$. With this formula, we can compute the initial distribution of every location according to the population data during a day. More generally, we can also apply this formula to the data of different seasons.

9.3.2 Realistic agents

9.3.2.1 Behaviors from deduction

In our model, the actions of agents are the results of a set of if-else statements. In the real work, we human make decision from thinking. In the future, we propose to study on how people make decision and then apply into the model to make it more reliable.

A promising approach here is BDI agents, which is expected to simulate the human behaviors.

9.3.2.2 Evacuation by vehicles

All our models are pedestrian evacuation, which means that people evacuate on foot. In fact, we have vehicles (bike, car, or also public transportation). The vehicles can provide a fast movement but it takes much place in the streets. In case of traffic jam, the vehicles become useless. More information is that the air transportation (helicopter) might be immune to traffic jam but it needs a place to land and takes which are hard to find in the city. Thus, the simulation of all these vehicles is a promising research direction. In our work, we make a relative achievement on simulation of buses and cars (more illustrations in appendix).

Appendix A

Illustration

A.1 Distribution representation for prediction survival rate by regression

A.1.1 Description

First, we focus on the nature of the evacuation. We realize that a certain sign placement changes the distribution of evacuees during the time. In fact, the evacuation is a problem of space and time. The fact that an evacuee is considered dead or alive totally depends on where and when he is at the end of the evacuation. The distribution of people at the end of the evacuation directly impacts to survival rate. Thus, our problem becomes how to build the feature describing the people distribution from the input sign placement.

In fact, if we make some signs in the real world, the signs would lead people in evacuation, which means that there are some places more crowded than others. Then we can focus only on some positions (or some vertices of interest) where people pass more frequently. These vertices are called beacons where we count the number of people. Let B_i be the distribution of beacon i at a certain moment of the evacuation (e.g. at the end of evacuation), then B_i is the percentage of people who are at vertex i at that moment. The feature vectors are built through 3 steps:

- initially evaluating the impact of each sign on people distribution: we run m simulations (m is number of arcs); each simulation takes only one arc as a sign placement; for each simulation, we get the people distribution on the predefined beacons. The result of this step is an initial beacon table (m x b) in which m is the number of lines corresponding number of arcs and b is the number of beacons.

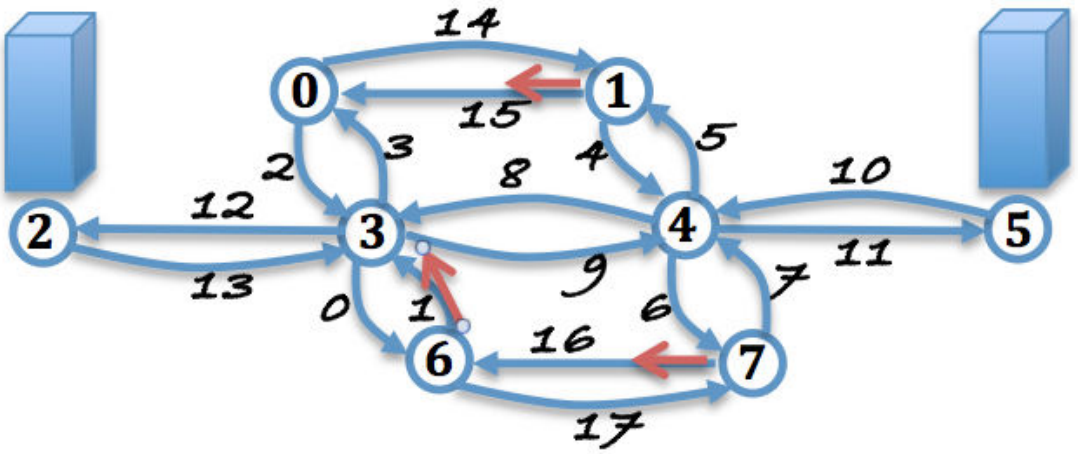


FIGURE A.1: An example graph of a block in the city

From our example described in figure A.1, if we take vertices (1, 2, 3, 4) as beacons, the result is presented in table A.1

- transforming sign placement into beacon distribution: from sign placement of K signs, we make a reference into the initial beacon table to get the people distribution of each arc on the predefined beacons; then, for an example, we have (b x K) values; If we build a data set of M example, we have a table of M lines (corresponding to M examples) and b x K columns. From the same example in figure A.1, for a sign placement of 3 signs (1, 15, 16), the table A.2 shows the data set of M examples.

- building feature data set: from the transforming step, for each beacon we have K values of distribution corresponding K signs in the example. In order to build the feature data set, we can implement one of these methods on K values of each beacon such as: sum, multiply, average, max, standard deviation, or even the ordered weighted averaging aggregation operator YAGER [1988]. The number of variables in the features is exactly the same number of beacons. For an example of average operator, for each beacon, we take the average of all the values. And we have 4 values corresponding 4 beacons. The table A.3 shows the data set built by making the average of K values for each beacon. Another example is presented in the table A.4 which shows the result from taking the max value.

Thus, with this approach about distribution, we have more than one method to build the data set (described below). Each method (operator) might have pros and cons. Which are the best candidate vertices for beacons, which are the best operator to build data set will be answered by experiment in the next sections.

	beacon 1	beacon 2	beacon 3	beacon 4
arc 0	0.1	0.2	0.1	0.3
arc 1	0	0.2	0.2	0.1
....	....	....	....	....
arc 17	0.3	0.4	0.1	0.1

TABLE A.1: Example of impact of single sign placement on distribution at predefined beacons

	beacon 1			beacon 2			beacon 3			beacon 4		
arc	1	15	16	1	15	16	1	15	16	1	15	16
ex 1	0.4	0.1	0.2	0	0.1	0.3	0.5	0.1	0	0.2	0.1	0.3
....	....	....	....	....	....	....	....	....	....	....	....	....
ex M	0.1	0.2	0.3	0.2	0.4	0	0.2	0.3	0.1	0.5	0.2	0

TABLE A.2: Example of data collected from 4 beacons (presented in the previous table)

	beacon 1	beacon 2	beacon 3	beacon 4	survival
example 1	0.23	0.13	0.2	0.2	0.3
....	....	....	....	....	....
example M	0.2	0.2	0.2	0.23	0.4

TABLE A.3: Example of data set from make average of K values of each beacon

	beacon 1	beacon 2	beacon 3	beacon 4	survival
example 1	0.4	0.3	0.5	0.3	0.3
....	....	....	....	....	....
example M	0.3	0.4	0.3	0.5	0.4

TABLE A.4: Example of data set from taking the max of K values of each beacon

- sum: we take the sum of all the distribution values of the K signs in the sign placement data.
- multiply: we multiply all the distribution values of the K signs together
- average: we take the average of all the distribution values
- max/min: we take the max/min value of the distribution values

A.1.2 Evaluation

In order to predict the survival rate, we use linear regression. For each instance (each placement of k signs), we compare the survival rate computed from simulation (called true survival) and survival come from prediction (called predicted survival). The true survival rate is computed by the same simulation in the previous chapters in which all parameters are reused. All cases of experimentation, number of signs is equal to 8% number of vertices (k=8%n).

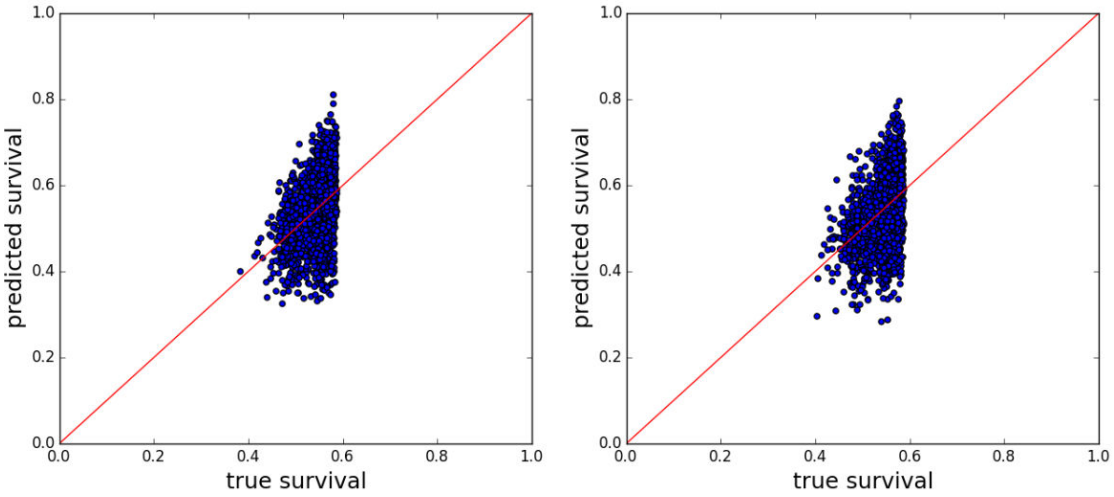


FIGURE A.2: Result of linear prediction of distribution representation, the feature are built from max values of distribution on beacons

Before making the quantitative evaluation, we must choose a measure. In this case we choose the coefficient of determination R^2 of the prediction (called score). Score is defined as (1 - u/v), where u is the regression sum of squares $\sum(yTrue - yPredict)^2$ and v is the residual sum of squares $\sum(yTrue - yTrue.mean())^2$. Best possible score is 1.0, lower values are worse.

A.1.2.1 Evaluation of operators to build features

The most important factor which impacts the performance of the prediction is the operator. In this case, we test one by one to find the best.

First, we analyze the result when we use max operator to compute the features. In this case, we take the maximum value for each beacon. The figure A.2 shows that the dots do not distributed around the secondary diagonal line, which means that the true survival rate and the predicted survival rate are not the same in most of the case. The conclusion is that max operator is not good. The reason for this result is that the distribution of each beacon is changed by the sign placement. Then, the maximum value of this beacon does not represent the distribution of each beacon in all cases. Since the max operator is not good, the min operator is not either.

Then, we test the average operator. In this case, the value of each beacon is computed by taking the average values of distribution on that beacon. The figure A.3 shows that the dots are around the secondary diagonal line, which means that the predicted survival rate is nearly the same as the true survival rate in most most of the case. This result is explained that the average value can represent the distribution of a beacon in all case

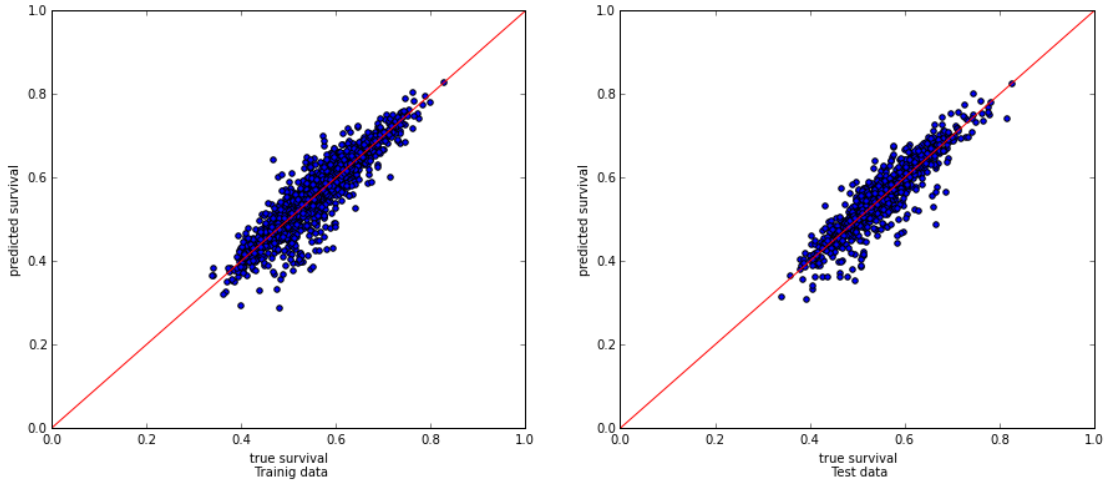


FIGURE A.3: Result of linear prediction of distribution representation, the feature are built from average values of distribution on beacons

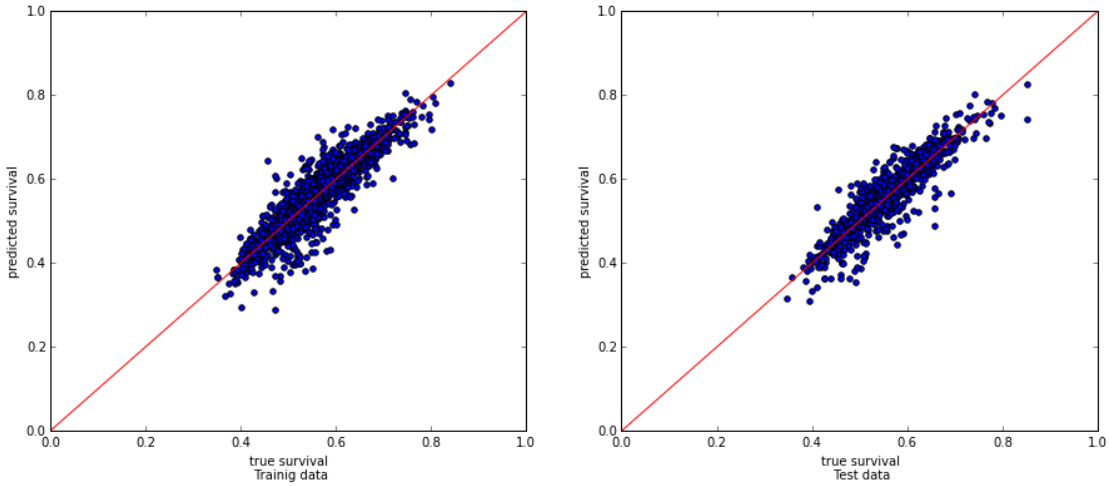


FIGURE A.4: Result of linear prediction of distribution representation, the feature are built by multiplying values of distribution on beacons

of the sign placement. Since the sum operator is similar to the average operator, the results from both cases are the same.

In the previous case, the average operator shows a positive result. We believe that the multiply operator is also good. In this case, we test the multiply operator. The value of each beacon is computed by multiplying all values of distribution on that beacon. The figure A.4 shows the result as good as the figure A.3, which means that both average operator and multiply operator outcome the good result. The reason for this result is that both operators accumulate all the values of all cases of sign placement.

Since the average operator and multiply operator are approximately good, we propose to concatenate them together. For each data set, we use all the method to build the feature data and then run the training and prediction phases to obtain the scores.

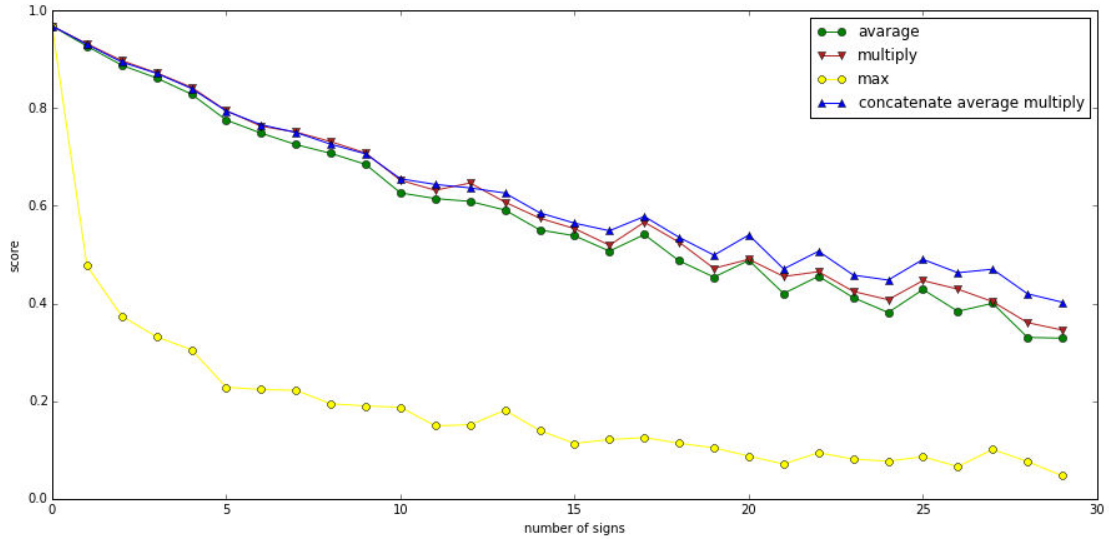


FIGURE A.5: Compare methods to build distribution features on train data set

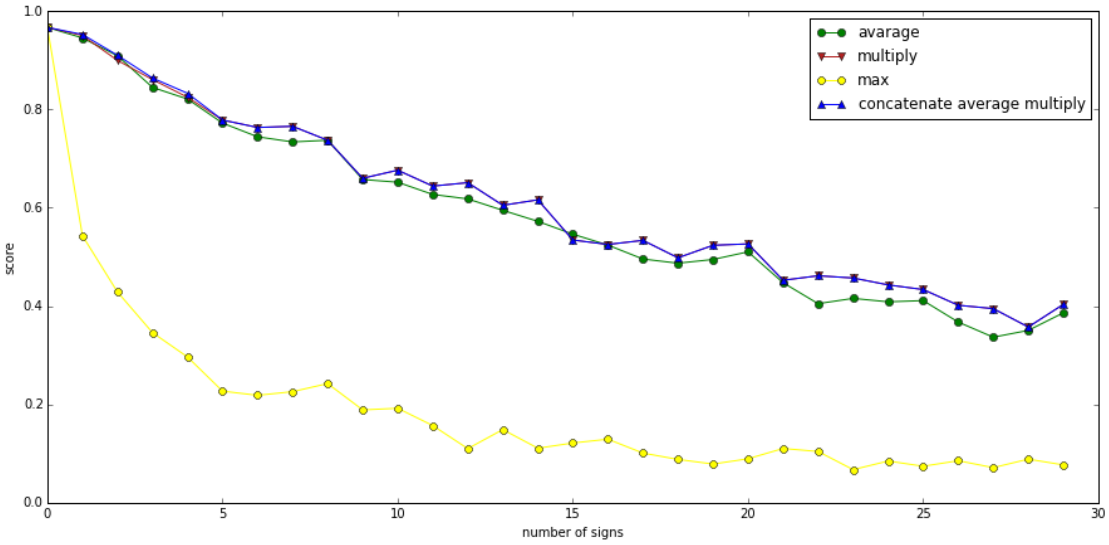


FIGURE A.6: Compare methods to build distribution features on test data set

After plotting the scores, the result (from figure A.5 and figure A.6) shows the comparison of all operators. While that max operator does not give good prediction, the average-multiply operator appears to be the best method for building features in case of distribution representation.

A.1.2.2 Evaluation: Shelter beacons vs non-shelter beacons

As describing in the previous section, if we choose the distribution representation, we have to indicate where are the beacons (or which vertices are used as beacons). In our representation, we have 2 types of vertices: the vertices near the safe places (high

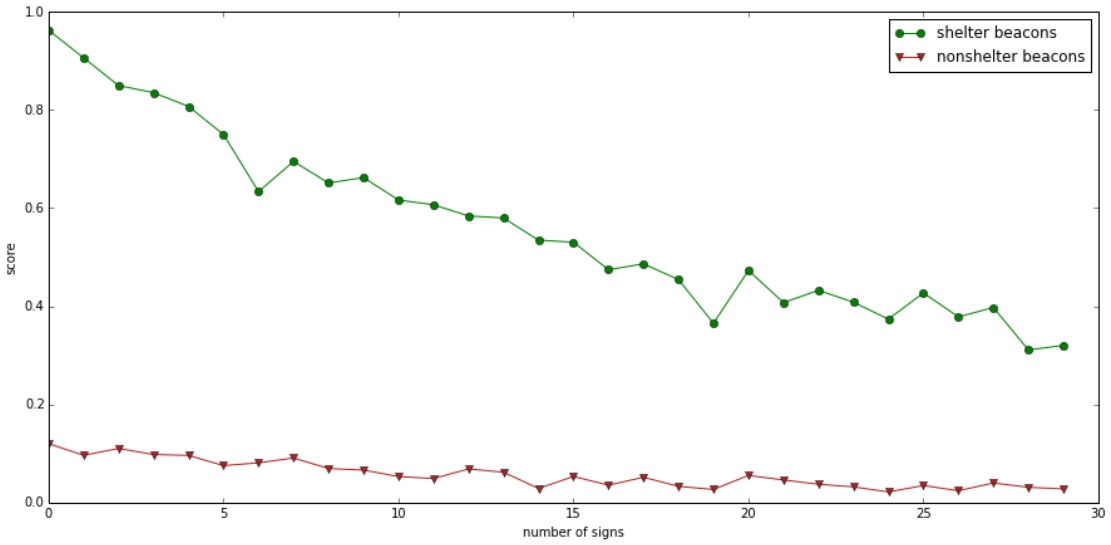


FIGURE A.7: Comparing scores of shelter beacons and non-shelter beacons on train data set

buildings or high ground positions) are called shelter vertices, and others are called non-shelter vertices. Then, we also have 2 types of candidate beacons: shelter beacons and non-shelter beacons.

In our problem, the ultimate goal of the signs is to lead evacuees to the shelters, then, the shelter beacons and non-shelter beacons would make the result of prediction different. In this experiment, we evaluate if shelter beacons are better than non-shelter beacons and how better they are. In this case, we implement 2 predictions: in the first prediction, the beacons are only the shelter vertices; and in the second one, the beacons are the random non-shelter vertices; the number of beacons in both cases are the same.

First, we run the evaluation of train data set. The survival rate of each sign placement is evaluated by simulation to obtain the label. The sign placement data are coded into 2 feature data sets by using the average values of distribution on beacons. In the first data set, all the beacons are shelter vertices, but the second data set is built from random non-shelter beacons. The distribution feature data are trained by linear regression. Then, the regression learner predicts the train data sets themselves to compute the scores. The figure A.7 shows the comparison of scores in both shelter and non-shelter beacons of distribution. In this figure, the shelter beacons are much better than non-shelter beacons on the train data.

We also evaluate the scores on the test data. We generate other data sets for test. We use the same linear learner (which is trained in the previous phase) to make prediction on the new data sets. The scores are noted that (in figure A.8) the shelter beacons are also better than non-shelter beacons. Then, we can conclude here that the shelter beacons are better then non-shelter beacons. The reason here is quite clear that the

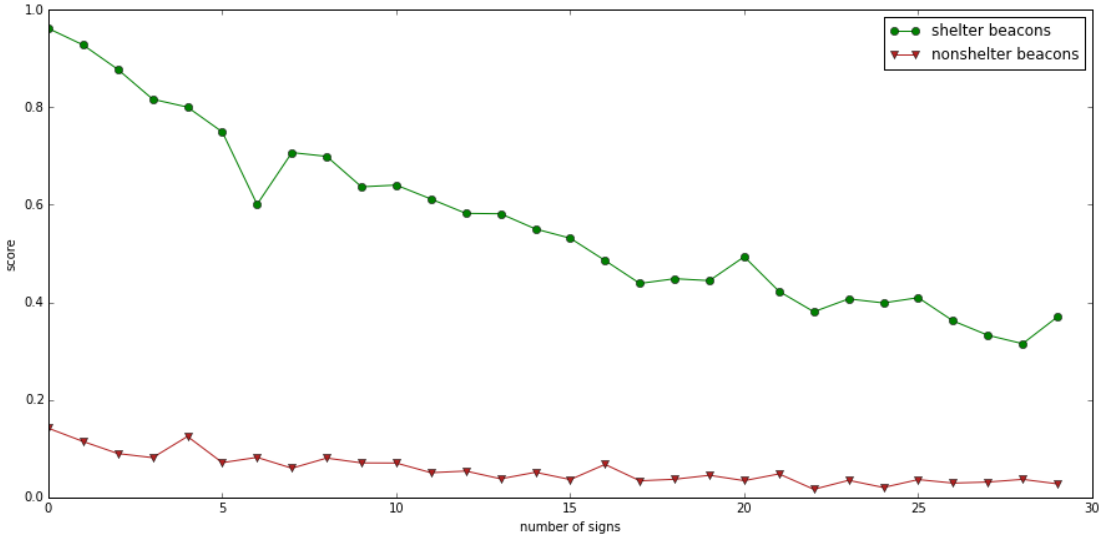


FIGURE A.8: Comparing scores of shelter beacons and non-shelter beacons on test data set

shelter vertices directly impact on survival rate because the evacuees arriving shelters are considered survivors. Therefore, from this moment, we only use shelter beacons.

Evaluation: impact of number of beacons on survival rate From the previous evaluation, the result shows that the shelter beacons are more important than non-shelter beacons. We also agree that adding more beacons makes the feature vectors longer. The learner then requires more computational resources to carry out the prediction (it would takes more time and also more memory to train and to predict). The question in this case is: does the scores get better if we add more beacons to the shelter beacons?

In order to verify this hypothesis, we first reuse all the data set generated for the previous experimentation. For each data set, we make 8 predictions in which the beacons are the shelter vertices and also adding specific number of random vertices. Concretely, in this case, we add gradually 10%, 20%, ..., 80% of vertices to the beacon set and compare then the results.

From experimentation on train data and test data, the results show that adding more beacons does not improve the score. The adding beacons might be good but they are redundant the shelter beacons. Then, we can only use the shelter vertices for beacons.

A.1.3 Conclusion on distribution representation

From the results of evaluation phase, we can conclude that:

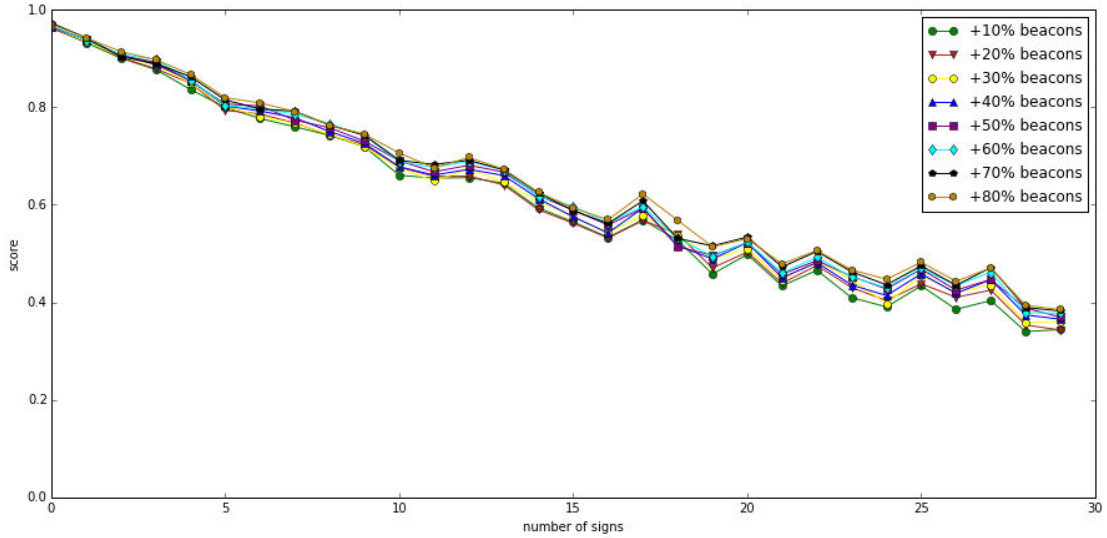


FIGURE A.9: Scores on train data sets obtained from adding more beacons

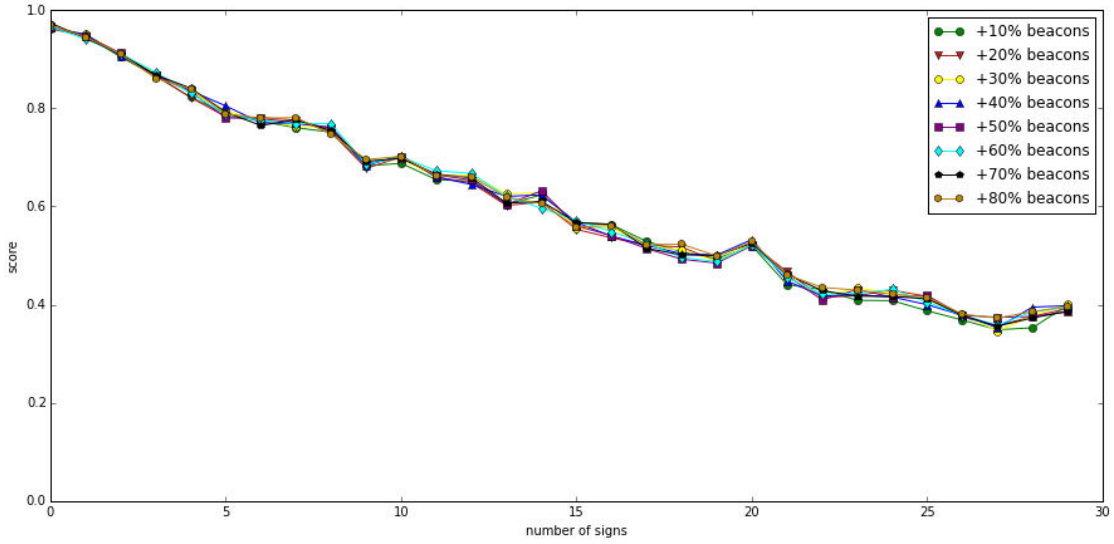


FIGURE A.10: Scores on test data sets obtained from adding more beacons

- Regarding the distribution representation, we have to transform the sign placement into feature vector.
- The combination of average operator and multiply operator is the best method to transform the sign placement to feature vector.
- The shelter vertices are the best choices to be beacons. The adding non-shelter vertices do not improve the results.
- Finally, in case we use distribution representation, the best solution is to use shelter vertices as beacons and to use combination of average operator and multiply operator to build features from sign placement.

While the distribution representation seems to be an acceptable solution, this representation itself opens a important question on the transforming phase from sign placement into feature vector: does the proposed method of transformation represent all the distribution of every vertex at all time of simulation? In the next section, we propose another approach to represent the features. We compare the results of both approaches to answer the question.

A.2 Simulation

In the previous sections, we focus on presentation of the results. Knowing that it would be better for the reader to see how our solution works, we make this appendix section. In this section, we would like to show all images of the interface of our system in order to illustrate our work.

The first appearance of the simulation is the environment. The environment in this case is the map of the city. Moreover, in our model, we enhance more information rather than simulating only a raw map. In figure A.11, the coastal area of the city are separated into different zones. Each zone is highlighted by a color representing the dangerous level. The red zone is certainly the most dangerous. Another informations are the buildings. In figure A.12, we can notice that there are 2 types of buildings: the normal buildings and the shelters.

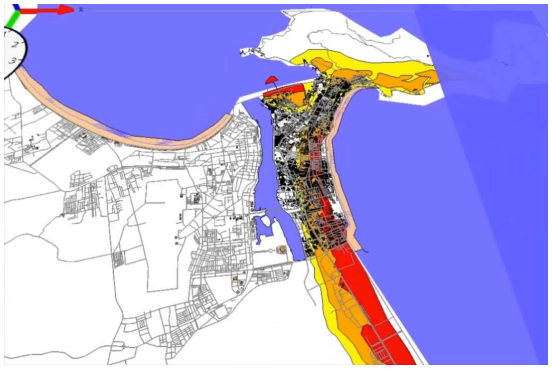


FIGURE A.11: Dangerous zones of coastal area of the city

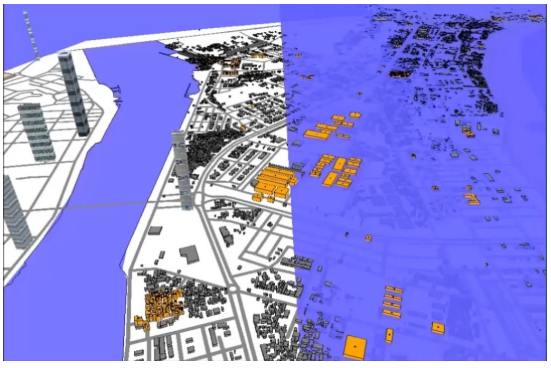


FIGURE A.12: Simulation Environment when tsunami is coming

The most important part of an Agent-Based Simulation are the agents. The agents here represent the pedestrians who move to the shelter during the simulation. Figure A.13 shows the agents (representing pedestrians) who are moving in the evacuation. Figure A.14 shows some agents who are very near a shelter. These agents are reaching shelter.

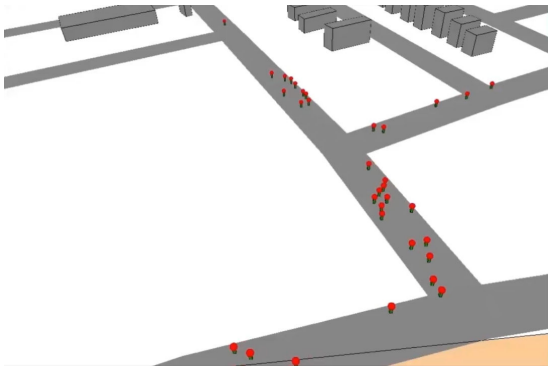


FIGURE A.13: Moving pedestrians

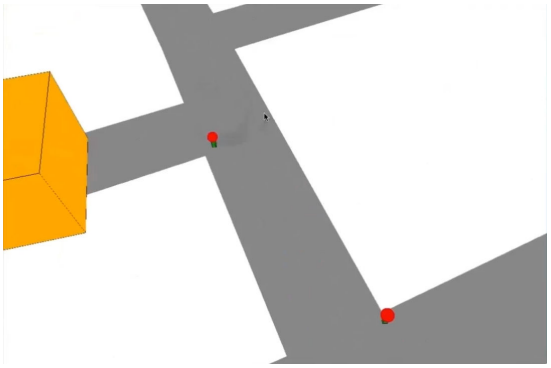


FIGURE A.14: Reaching shelter pedestrians

The most important part of this problem is the signs. While figure A.15 shows only a sign with agents following its direction, figure A.16 presents a group of signs which are successive one by one.

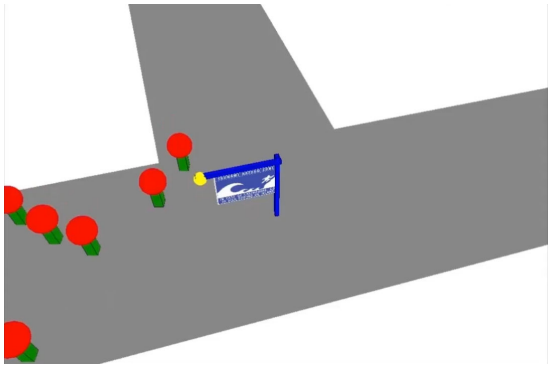


FIGURE A.15: Sign with nearby agents

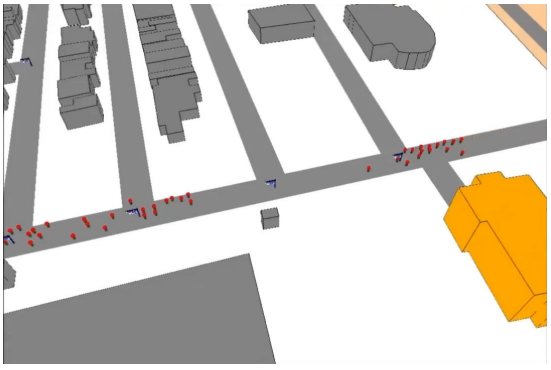


FIGURE A.16: Successive signs leading agent in evacuation

A.3 Graphich User Interface

In our system, we provide a simple way for user to intervene the optimization. In this case, the user can place some signs. The system re-optimizes their direction and than find the optimization of other signs. The figure A.17 shows a function which permits user placing a sign on a certain street. In the figure A.18 the pre-defined signs are re-directed by the optimization system.

A.4 Work in progress

We are aware of the fact that there are some vehicles used in evacuation. In fact, a bus might be useful to carry children and old people who can not walk a long distance, which

FIGURE A.17: User place a sign by hand

FIGURE A.18: System re-optimize the directions and find other signs

motivates use to build vehicle agents. In our work, we are building a simulation of the movement of buses and cars. Figure A.19 and figure A.20 show the current achievement of our vehicles model. While this model is still simple, it opens a new direction of research which might lead us to a huge application area such as: simulation of urban traffic jam or problem of optimization of bus lines in a bus system.

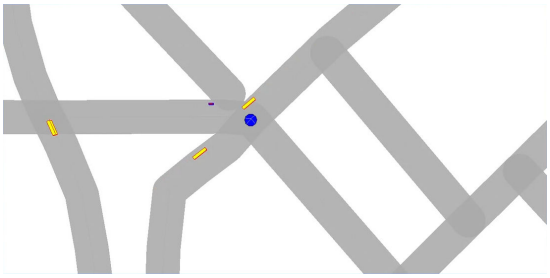


FIGURE A.19: Demo 1 of simulation of buses and cars

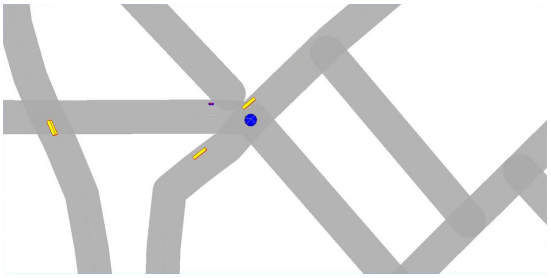


FIGURE A.20: Demo 2 of simulation of buses and cars

Bibliography

BACK, T. (1996). Evolutionary algorithms in theory and practice.

BARTON, R. R. (2009). Simulation optimization using metamodels. In *Winter Simulation Conference*, pages 230–238. Winter Simulation Conference.

BARTON, R. R. and MECKESHEIMER, M. (2006). Metamodel-based simulation optimization. *Handbooks in operations research and management science*, 13:535–574.

BELTAIEF, O., HADOUAJ, S., and GHEDIRA, K. (2011). Multi-agent simulation model of pedestrians crowd based on psychological theories. In *In Proc. the 4th International Conference on Logistics*, pages 150–156.

BOUSQUET, O. and BOTTOU, L. (2008). The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168.

BRANKE, J., GRECO, S., SŁOWIŃSKI, R., and ZIELNIEWICZ, P. (2009). Interactive evolutionary multiobjective optimization using robust ordinal regression. In *Evolutionary multi-criterion optimization*, pages 554–568. Springer.

CHOW, W. K. (2007). Waiting time for evacuation in crowded areas. *Building and Environment*, 42(10):3757–3761.

CHRISTENSEN, K. (2008). Agent-Based Emergency Evacuation Simulation with Individuals with Disabilities in the Population. *Journal of Artificial Societies and Social Simulation*, 11(3).

CHURCH, R. L. and COVA, T. J. (2000). Mapping evacuation risk on transportation networks using a spatial optimization model. *Transportation Research Part C: Emerging Technologies*, 8(1):321–336.

COZAD, A., SAHINIDIS, N. V., and MILLER, D. C. (2014). Learning surrogate models for simulation-based optimization. *AIChE Journal*, 60(6):2211–2227.

DALL'OSSO, F. and DOMILEY-HOWES, D. (2010). Public assessment of the usefulness of" draft" tsunami evacuation maps from sydney, australia–implications for the

establishment of formal evacuation plans. *Natural Hazards and Earth System Sciences*, 10(8):1739–1750.

D'AMBROSIO, C. and LODI, A. (2011). Mixed integer nonlinear programming tools: a practical overview. *4OR*, 9(4):329–349.

DANIELSEN, F., SORENSEN, M. K., OLWIG, M. F., SELVAM, V., PARISH, F., BURGESS, N. D., HIRAISHI, T., KARUNAGARAN, V. M., RASMUSSEN, M. S., HANSEN, L. B., et al. (2005). The asian tsunami: a protective role for coastal vegetation. *Science(Washington)*, 310(5748):643.

DROGOUL, A., AMOUROUX, E., CAILLOU, P., GAUDOU, B., GRIGNARD, A., MARILLEAU, N., TAILLANDIER, P., VAVASSEUR, M., VO, D.-A., and ZUCKER, J.-D. (2013). GAMA: Multi-level and Complex Environment for Agent-based Models and Simulations (demonstration). In *the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1361–1362.

DUBOZ, R., VERSMISS, D., TRAVERS, M., RAMAT, E., and SHIN, Y. J. (2010). Application of an evolutionary algorithm to the inverse parameter estimation of an individual-based model. *Ecological modelling*, 221(5):840–849.

FANG, Z., LO, S. M., and LU, J. A. (2003). On the relationship between crowd density and movement velocity. In *Fire safety journal*, volume 38, pages 271–283.

FESTINGER, L. (1957). *A theory of social comparison processes*. Stanford University Press.

FRANZ, R. (2006). Representations for genetic and evolutionary algorithms.

FRIDMAN, N. and KAMINKA, G. (2007). Towards a cognitive model of crowd behavior based on social comparison theory. In *AAAI*.

GONZALEZ, F. I., GEIST, E. L., JAFFE, B., KANOGLU, U., MOFIELD, H., SYNOLAKIS, C. E., TITOV, V. V., ARCAS, D., BELLOMO, D., CARLTON, D., HORNING, T., JOHNSON, J., NEWMAN, J., PARSONS, T., PETERS, R., PETERSON, C., PRIEST, G., VENTURATO, A., WEBER, J., WONG, F., and YALCINER, A. (2009). Probabilistic tsunami hazard assessment at Seaside, Oregon, for near- and far-field seismic sources. *Journal of Geophysical Research*, 114(C11):1–19.

GOSAVI, A. (2014). *Simulation-based optimization: parametric optimization techniques and reinforcement learning*, volume 55. Springer.

GOTO, Y., AFFA, M., AUGUSSABTI, NURDIN, Y., YULIANA, D. K., and ARDIANSYAH (2012). Tsunami Evacuation Simulation for Disaster Education and City Planning. *Journal of Disaster Research*, 7(1):92–101.

HANSEN, N. (2006). The cma evolution strategy: a comparing review. In *Towards a new evolutionary computation*, pages 75–102. Springer.

HANSEN, N. and KERN, S. (2004). Evaluating the cma evolution strategy on multimodal test functions. In *International Conference on Parallel Problem Solving from Nature*, pages 282–291. Springer.

HELBING, D. and MOLNAR, P. (1995). Social force model for pedestrian dynamics. In *Phys Rev E 1995, 51(5)*, pages 4282–4286.

HENDERSON, S. G. and NELSON, B. L. (2006). *Handbooks in Operations Research and Management Science: Simulation*, volume 13. Elsevier.

HERRERA, F., LOZANO, M., and VERDEGAY, J. L. (1998). Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial intelligence review*, 12(4):265–319.

IMAMURA, F., MUHARI, A., MAS, E., PRADONO, M. H., POST, J., and SUGIMOTO, M. (2012). Tsunami disaster mitigation by integrating comprehensive countermeasures in padang city, indonesia. *J Disaster Res*, 7(1):48–64.

INGIMUNDARDOTTIR, H. and RUNARSSON, T. P. (2011). Sampling strategies in ordinal regression for surrogate assisted evolutionary optimization. In *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pages 1158–1163. IEEE.

JI, Q. and GAO, C. (2007). Simulating crowd evacuation with a leader-follower model. In 1, editor, *International Journal of Computer Sciences and Engineering Systems*, volume 4, pages 27–30.

KASAHARA, H., MORI, M., KURUMATANI, K., MUKUNOKI, M., and MINOH, M. (2014). A tourism information service for safety during school trips. In *Serviceology for Services*, pages 319–326. Springer.

KATADA, T. and KANAI, M. (2008). Implementation of tsunami disaster education for children and their parents at elementary school. *Solutions to coastal disasters*.

KUROWSKI, M. J., HEDLY, N., and CLAGUE, J. J. (2011). An assessment of educational tsunami evacuation map designs in washington and oregon. *Natural hazards*, 59(2):1205–1223.

LE, V.-M., ADAM, C., CANAL, R., GAUDOU, B., HO, T. V., and TAILLANDIER, P. (2010). Simulation of the Emotion Dynamics in a Group of Agents in an Evacuation Situation. In *In the first Pacific Rim workshop on Agent-based modeling and simulation of Complex Systems (PRACSYS), included into the The 13th International*

Conference on Principles and Practice of Multi-Agent Systems (PRIMA), Calcutta, India, included into the The 13, pages 604–619.

LI, G., ZHOU, Y., and LIU, M. (2014). Comprehensive optimization of emergency evacuation route and departure time under traffic control. *The Scientific World Journal*, 2014.

LIN, P., LO, S. M., HUANG, H. C., and YUEN, K. K. (2008). On the use of multi-stage time-varying quickest time approach for optimization of evacuation planning. *Fire Safety Journal*, 43(4):282–290.

LIU, Y., OKADA, N., SHEN, D., and LI, S. (2010). Agent-based Flood Evacuation Simulation of Life- threatening Conditions Using Vitae System Model. *Journal of Natural Disaster Science*, 31(2):33–41.

LIU, Y., OKADA, N., and TAKEUCHI, Y. (2008). Dynamic Route Decision Model-based Multi-agent Evacuation Simulation - Case Study of Nagata Ward, Kobe. *Journal of Natural Disaster Science*, 28(2):91–98.

LIU, Y., WANG, W., HUANG, H. Z., LI, Y., and YANG, Y. (2014). A new simulation model for assessing aircraft emergency evacuation considering passenger physical characteristics. *Reliability Engineering & System Safety*, 121:187–197.

LONG, B., CHAPELLE, O., ZHANG, Y., CHANG, Y., ZHENG, Z., and TSENG, B. (2010). Active learning for ranking through expected loss optimization. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274. ACM.

MACAL, C. M. and NORTH, M. J. (2011). Introductory tutorial: agent-based modeling and simulation. In *The 2011 Winter Simulation Conference*, pages 1456–1469.

MAS, E., IMAMURA, F., and KOSHIMURA, S. (2012). An agent based model for the tsunami evacuation simulation. A case study of the 2011 great east japan tsunami in Arahama town. In *Joint conference proceedings 9th International Conference on Urban Earthquake Engineering/4th Asia Conference on Earthquake Engineering*, pages 1957–1964.

MAURER, A. and PONTIL, M. (2009). Empirical bernstein bounds and sample variance penalization. In *Proc. Int. Conf. Learn. Theory, 2009*, pages 1–9.

METOYER, R. and HODGINS, J. (2004). Reactive pedestrian path following from examples. In 20, editor, *The Visual Computer*, volume 10, pages 635–649.

MINAMOTO, T., NARIYUKI, Y., FUJIWARA, Y., and MIKAMI, A. (2008). Development Of Tsunami Refuge Petri-Net Simulation System Utilizable In Independence Disaster Prevention Organization. In *The 14th World Conference on Earthquake Engineering October 12-17, 2008, Beijing, China*.

MUSSE, S. R. and THALMANN, D. (1997). A Model of Human Crowd Behavior : Group Inter-Relationship and Collision Detection Analysis. In *Workshop of Computer Animation and Simulation of Eurographics*, pages 39–51.

NGUYEN, H. P., VU, H. P., and PHAM, T. T. (2012a). Development of a tsunami evacuation plan for urban area of Nhatrang city using GIS. *Journal of Earth's sciences (in Vietnamese)*.

NGUYEN, M. H., HO, T. V., NGUYEN, T. N. A., and ZUCKER, J. D. (2012b). Which Behavior Is Best in a Fire Evacuation : Simulation with the Metro Supermarket of Hanoi. In *International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, pages 1–6.

NGUYEN, T. N. A., CHEVALEYRE, Y., and ZUCKER, J. D. (2011). Optimizing the Placement of Evacuation Signs on Road Network with Time and Casualties in case of a Tsunami. In *20th IEEE International Conference on Collaboration Technologies and Infrastructures (WETICE 2011)*, number i, pages 394–396.

NGUYEN, T. N. A., ZUCKER, J. D., NGUYEN, H. D., DROGOUL, A., and VO, D. A. (2012c). A hybrid equation-based and agent-based modeling of crowd evacuation on road network. In *Proc. ICCS*, pages 456–466.

NGUYEN, T. N. A., ZUCKER, J. D., NGUYEN, M. H., DROGOUL, A., and NGUYEN, H. P. (2012d). Simulation of emergency evacuation of pedestrians along the road networks in Nhatrang city. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference*, pages 1–6.

PARK, S., VAN DE LINDT, J. W., GUPTA, R., and COX, D. (2012). Method to determine the locations of tsunami vertical evacuation shelters. *Natural hazards*, 63(2):891–908.

PARUNAK, H., SAVIT, R., and RIOLO, R. (1998). Agent-based modeling vs equation-based modeling: A case study and users' guide. In *Sichman, J., S., Conte, R., Gilbert, N. (eds.) Multi-Agent Systems and Agent-Based Simulation, Lecture Notes in Artificial Intelligence*, pages 10–25.

PELECHANO, N. and N.I., B. (2006). Modeling crowd and trained leader behavior during building evacuation. In *IEEE Computer Graphics and Applications*, volume 26, pages 80–86.

PELECHANO, N., OBRIEN, K., and BADLER, N. (2005). Crowd simulation incorporating agent psychological models, roles and communication. In *In Proc. the 1st International Workshop on Crowd Simulation*, pages 21–30.

PÉROCHE, M., LEONE, F., and GUTTON, R. (2014). An accessibility graph-based model to optimize tsunami evacuation sites and routes in martinique, france. *Advances in Geosciences*, 38(38):1–8.

QIU, F. and HU, X. (2010). Modeling group structure in pedestrian crowd simulation. In 2, editor, *Simulation Modelling Practice and Theory*, volume 18, pages 190–205.

ROGSCH, C., GALSTER, R., LUTHARDT, T., and MOHR, D. (2014). The effect of pedestrian placement and pre-movement times on evacuation simulation. *Transportation Research Procedia*, 2:291–299.

RONCHI, E. and NILSSON, D. (2013). Fire evacuation in high-rise buildings: a review of human behaviour and modelling research. In *Fire Science Reviews 2:7*.

ROS, R. and HANSEN, N. (2008). A simple modification in cma-es achieving linear time and space complexity. In *International Conference on Parallel Problem Solving from Nature*, pages 296–305. Springer.

SAITO, T. and KAGAMI, H. (2004). Simulation Of Evacuation Behavior From Tsunami Utilizing Multi Agent System. In *13th World Conference on Earthquake Engineering, Vancouver, B.C., Canada, August 1-6, 2004*, number 612, page 612.

SAITTA, L. and ZUCKER, J. D. (2013). *Abstraction in artificial intelligence and complex systems*. Springer.

SCHEER, S., GARDI, A., GUILLANDE, R., EFTICHIDIS, G., VARELA, V., VANSSAY, B. D., and COLBEAU-JUSTIN, L. (2011a). *Handbook of Tsunami Evacuation Planning*. Publications Office of the European Union.

SCHEER, S., GUILLANDE, R., and GARDI, A. (2011b). Optimizing Tsunami Evacuation Plans Through the Use of Damage Scenarios. In *Earthzine no. Theme Issue 2011(1). Disaster Management*, pages 1–13.

SETTLES, B. (2010). Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

SHAO, W. and TERZOPOULOS, D. (2005). Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH Eurographics symposium on Computer animation*.

SHIFLET, A. and SHIFLET, G. (2014). An introduction to agent-based modeling for undergraduates. In *Procedia Comput. Sci. 29*, pages 1392–1402.

SHIRAZI, A. S., DAVISON, T., VON MAMMEN, S., DENZINGER, J., and JACOB, C. (2014). Adaptive agent abstractions to speed up spatial agent-based simulations. *Simulation Modelling Practice and Theory*, 40:144–160.

SILVA, T., WUWONGSE, V., and SHARMA, H. N. (2011). Linked data in disaster mitigation and preparedness. In *Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on*, pages 746–751. IEEE.

SMITH, J. L. and BROKOW, J. T. (2008). Agent Based Simulation of Human Movements during Emergency Evacuations of Facilities. In *Structures Congress 2008*, pages 1–10, Reston, VA. American Society of Civil Engineers.

STONEDAHL, F. and WILENSKY, U. (2010). Finding Forms of Flocking : Evolutionary Search in ABM Parameter-Spaces. In *MABS 2010*, pages 61–75.

SUPPASRI, A., IMAMURA, F., and KOSHIMURA, S. (2012). Tsunami Hazard and Casualty Estimation in a Coastal Area That Neighbors the Indian Ocean and South China Sea. *Journal of Earthquake and Tsunami*, 06(02):1–25.

TANAKA, N. (2009). Vegetation bioshields for tsunami mitigation: review of effectiveness, limitations, construction, and sustainable management. *Landscape and Ecological Engineering*, 5(1):71–79.

TANIOKA, Y., LATIEF, H., SUNEDAR, H., GUSMAN, A. R., and KOSHIMURA, S. (2012). Tsunami Hazard Mitigation at Palabuhanratu, Indonesia. *Journal of Disaster Research*, 7(1):19–25.

TITOV, V. V., GONZALEZ, F. I., BERNARD, E. N., EBLE, M. C., MOFJELD, H. O., NEWMAN, J. C., and VENTURATO, A. J. (2005). Real-Time Tsunami Forecasting: Challenges and Solutions. *Nat. Hazards*, 35(206):45–58.

TSAI, J., FRIDMAN, N., BOWRING, E., BROWN, M., EPSTEIN, S., KAMINKA, G., MARSELLA, S., OGDEN, A., RIKI, I., SHEEL, A., TAYLOR, M., WANG, X., TAMBER, M., and ZILKA, A. (2011). Escapes - evacuation simulation with children, authorities, parents, emotions, and social comparison. In *In Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems AAMAS 2011*, pages 457–464.

VORST, H. C. M. (2010). Evacuation models and disaster psychology. *Procedia Engineering*, 3:15–21.

VU, T. C. and NGUYEN, D. X. (2008). Tsunami risk along Vietnamese coast. *Journal of Water Resources and Environmental Engineering*, (23):24–33.

WEISE, T. (2009). Global optimization algorithms-theory and application. *Self-Published,*.

WICKELGREN, W. A. (1977). Speed-accuracy tradeoff and information processing dynamics. *Acta psychologica*, 41(1):67–85.

YAGER, R. R. (1988). On ordered weighted averaging aggregation operators in multi-criteria decision making. In *IEEE Transactions on Systems, Man and Cybernetics 18*, pages 183–190.