Delft University of Technology
Faculty Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Understanding Terrorist Activity: Is Agent-Based Modelling a viable solution?

(Dutch title: Inzicht krijgen in terrorisme: Is een Agent-Gebaseerd Model een uitvoerbare oplossing?)

Report for the purpose of
Delft Institute of Applied Mathematics
as part to obtain

the degree of

BACHELOR OF SCIENCE
in
APPLIED MATHEMATICS

by

DAVEY KAAK

Delft, the Netherlands
June 2017

BSc report APPLIED MATHEMATICS

"Understanding Terrorist Activity: Is Agent-Based Modelling a viable solution?"

(Dutch title: "Inzicht krijgen in terrorisme: Is een Agent-Gebaseerd Model een uitvoerbare oplossing?")

DAVEY KAAK

Delft University of Technology

Supervisor

Dr. P. Cirillo

Other members of the committee

Dr. Ir. W.G.M. Groenevelt

Dr. J.G. Spandaw

June, 2017

Delft

Abstract

In this report an Agent-Based Model created by Bulleit and Drewek used for analysing terrorist attacks will be implemented and compared with reality.

First the implementation of the model is explained based on different articles written by Bulleit and Drewek. After that the data of the simulations are analysed statistically. In this analysis the model is compared with the Global Terrorism Database for consistency with historical data and the model is compared for consistency in distributions of the data based on research reports. Finally, a few extensions will be implemented and again analysed statistically.

An Agent-Based Model for terrorist activity can give us insights in the behaviour of terrorists. It can especially be used for security policies or when we want to investigate what happens with an implementation of a new type of security.

Contents

Chapter 1

Introduction

Unfortunately quite a number of terrorist events have happened over the past months. Manchester Arena bombing, Stockholm truck attack, Berlin Christmas market attack, Nice truck attack, Brussels bombings and November Paris attacks are only a few examples of the many attacks that have happened so far [3]. Many actions have been taken over time in order to defend ourselves against terrorists and terrorist attacks, De Graaf indicates in her talk that the police and intelligence services need to do everything in order to keep (or get) ahead of terrorists [14]. Figure 1.1 shows that the number of self-destructing attacks worldwide (also called bombings) per year has seen an extreme upward spike in the years 2001 till 2015.

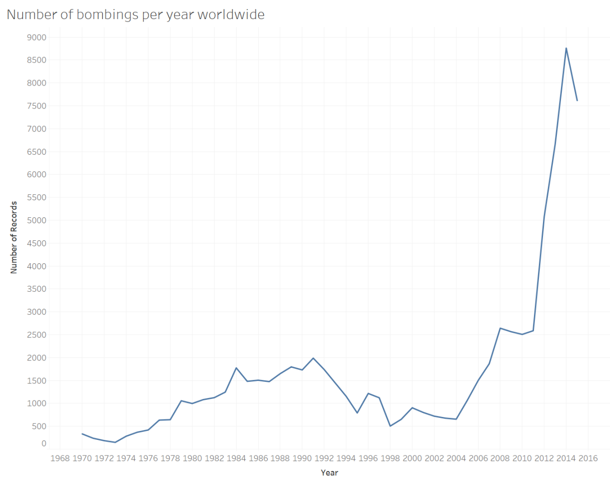

Figure 1.1: Number of self-destructing attacks (or bombings) per year according to the Global Terrorism Database [29]. As it can be seen, between 2001 and 2015 there was an extreme upward spike in the number of bombings per year.

It is therefore of importance to focus on ways to fight the number of self-destructing attacks. For example De Graaf mentions that a day to day routine is gathering intelligence in the form of phone taps and internet posts or information gathered by undercover agents [14]. But besides intelligence in the form of real-time data, one can also attempt to analyse historical data and try to see underlying patterns. Chen et al. [11], Santiford-Jordan [32] and Sandler [31] are a few examples of researchers that analysed historical data related to terrorist activities. As Tutun, Khasawneh, and Zhuang [38, p.358] mention: "Predicting suicide attacks, which encompasses high uncertainty, is almost impossible. The uncertain nature of terrorism is the main challenge in the design of counter-terrorism policy." It is therefore hard to make a good model that predicts when and where terrorist attacks will happen. Instead of predicting attacks based on historical evidence, one can also analyse the general way terrorists behave. By analysing this the authorities can get an idea where terrorists are most likely to plan an attack and therefore determine where security is most needed. Bulleit and Drewek [5] are one of the few that have created a model to analyse terrorist activity. They have used the way people act in order to create a model that represents a (small) society. This modelling technique is often used in Economics under the name of Agent-Based Modelling.

1.1 Terrorism

Before we start to get into detail about Agent-Based Modelling, let us first focus on the basic question: What actually is 'terrorist activity'? Or similarly, what is 'terrorism'? Someone at a police station might have a very different meaning for this phrase compared to a civilian in society. This is actually also the reason why states are troubled by defining one standard definition for terrorism. Schmid [35] and Walter [41] both concluded the same in their analysis of the various definitions of terrorism. They conclude that there are such big differences among the definitions that no consensus can be made. It is not necessary for this research to choose one definition over the other, but it should be noted that there are various definitions used in this context. The Global Terrorism Database [28, p.9] gives as definition: "The threatened or actual use of illegal force and violence by a non-state actor to attain a political, economic, religious, or social goal through fear, coercion, or intimidation." In this report we will stick to this definition.

1.2 Agent-Based Modelling

Agent-Based Modelling (ABM) is a computational technique to model the behaviour of individuals and analyse their effect on the society that has been modelled. ABM is often used to model complex systems, which are difficult to study analytically. Agent-Based Models are said to generally have two properties [2]:

- They are based on interactions among agents;
- Effects that are caused by these interactions cannot be analysed by only addressing the individual agents separately.

The advantages of Agent-Based Modelling over other types of analysis are that we can easily change the initial conditions and therefore analyse different set-ups for the same model. Furthermore, an Agent-Based Model is a simplification of reality, making it a simulation that is a lot faster than one specific event in real life. Moreover, Agent-Based Models can be simulated multiple times in order to check for consistency of the outcomes [22].

In other literature Agent-Based Modelling (ABM) is also named Individual-Based Modelling (IBM), Agent-Based Modelling and Simulation (ABMS) or Agent-Based Simulation (ABS) [25]. For this research the term Agent-Based Modelling is used.

Agent-Based Modelling has been used numerous times in research. Schelling [34], Axelrod [1], Cederman [9][10], Epstein and Axtell [17] and Macy and Willer [26] are named by Bulleit and Drewek [5] as examples of other researches in which Agent-Based Modelling has been used for specific or general situations. Bulleit and Drewek [5] have made an Agent-Based Model that analyses terrorist attacks. The main aim of their model is to predict where terrorist agents are most likely perform an attack.

The analysis Bulleit and Drewek did on their model [5] stated that in an initial validation the attack magnitudes are power law distributed and that attacks are planned in regions with high wealth, or where high wealth passes by. The analysis is based on a few statistical tests and some visual interpretation. The results of the initial validation and analysis gives the idea of a generally good model, and we will therefore compare the model with the Global Terrorism Database on the consistency with historical data.

It should be noted that since the model in this project is based on a description of the model by Bulleit and Drewek and not the original code the model described in this project is only partially based on the model of Bulleit and Drewek of 2011. The model used in this research is mainly based on the model explained by Bulleit and Drewek in *Agent-Based Simulation for Human-Induced Hazard Analysis* (2011) [5] and for some details *Simulating Initial Conditions in Agent-Based Modeling* (2005) [6] is used.

Furthermore, we should note that Bulleit and Drewek have published various articles on the Agent-Based Model they created. *Simulating Initial Conditions in Agent-Based Modeling* [6], *An Agent-Based Model of Terrorist Activity* [7], *Simulating Terrorism in a Community* [8], *Agent-Based Simulation for Human-Induced Hazard Analysis*[5] and *Agent-Based Modeling and Simulation for Hazard Management* [4] are only a few examples.

Between these various articles and reports published by Bulleit and Drewek there are minor differences in the models presented. These differences will likely be the case due to the different circumstances in which the model is used. But all the articles explain situations in which terrorist activity is analysed, it is therefore of importance to state that the main article of Bulleit and Drewek used in this report is one of the many articles that they have written and that these other articles might lead to small differences in the conclusions.

The main article *Agent-Based Simulation for Human-Induced Hazard Analysis* [5] that has been used for this research has been chosen since it was the most detailed report available at the start of this research. Furthermore, other articles of Bulleit and Drewek refer to this report for certain details, which makes the model explained in this report to be believed as some basis for the other models.

1.3 Global Terrorism Database

As is already indicated, the main goal of this research is to check the consistency of historical data with the model described by Bulleit and Drewek. To check this consistency we will make use of the Global Terrorism Database (GTD) [29].

The Global Terrorism Database is a database that contains all worldwide attacks that are believed to be terrorist attacks according to the definition of the GTD [28]. Per attack several information variables are stored, for this research especially the information variables on the number of attacks and the number of casualties will be used. The GTD dataset starts in the year 1970 and contains the terrorist attacks until 2015. As is noted in the Codebook [28] only the year 1993 is unavailable.

1.4 Overview of the research

The research in this project tries to combine two of the fields discussed before. The main goal is to compare the model of Bulleit and Drewek with the historical data of the Global Terrorism Database on order to get an idea whether or not an implementation of the model of Bulleit and Drewek is representative of reality. We will do this by starting to analyse two papers of Bulleit and Drewek in chapter 2 in order to implement a model in Matlab [37]. In Chapter 3 the results of this model will be analysed and compared to the Global Terrorism Database. Distributions used in this chapter are shortly explained in Appendix A (page 63). In chapter 4 some extensions to this model will be proposed and their results will be analysed. Finally the conclusions based on this research will be presented and discussed. All code that has been used for the different models explained in this report can be found in Appendices B-D (page 65).

Chapter 2

State of the art: Bulleit and Drewek

As stated in the introduction, the model used in this research is mainly based on the model explained by Bulleit and Drewek in *Agent-Based Simulation for Human-Induced Hazard Analysis* (2011) [5] and for some details *Simulating Initial Conditions in Agent-Based Modeling* (2005) [6] is used. Using these reports we will first explain the implementation of the environment in paragraph 2.1 and after that the implementation of the different agents. Finally, in paragraph 2.3 the differences with respect to the original main model of Bulleit and Drewek will be explicitly stated and explained as well as some additional comments on the model of Bulleit and Drewek. The code that relates to the explanations in this chapter can be found in Appendix B (page 65).

2.1 Environment

The environment is a 50×50 grid, on which a community is represented by 9 resource piles and 500 civilian agents at time t_0. The resources piles are classified by 4 different resources, which can be thought of e.g. home, work, shops and sports facilities. Furthermore, the area is a walled environment, meaning that agents cannot move outside the boundaries. The set-up of the community is well represented by the figure 2.1, which is originally included in the article of Bulleit and Drewek [5].

As Bulleit and Drewek state in their article: "The terrorist hazard could be analysed in a manner similar to the way that seismic hazard is handles"[5, p.205]. This is also one of the reasons why the model is based upon the SugarScape type model. The SugarScape model is used to define the community in which the agents live. In accordance with the SugarScape type model the model makes 9 distinct piles for each resource [17]. The value on each node ranges between 1.0 and 30.0, and is built up like a sugar pile, meaning that the centre of a resource pile has value 30.0 and the side has value 1.0. Nodes on which no resource pile is situated have value 0.0. At every timestep the value of the resource piles grow back with $\frac{1}{4}$ of the original value at t_0 $(n = 0)$. The maximum value of each node in the resource pile is the value assigned at t_0. We therefore get the following equation for every node

$$R_n = \begin{cases} R_{n-1} + \frac{1}{4} \cdot R_0 & \text{if } R_{n-1} < \frac{3}{4}R_0 \\ R_0 & \text{if } R_{n-1} \geq \frac{3}{4}R_0 \end{cases} \tag{2.1}$$

in which R_n is the resource value at the node at time $n \geq 1$.

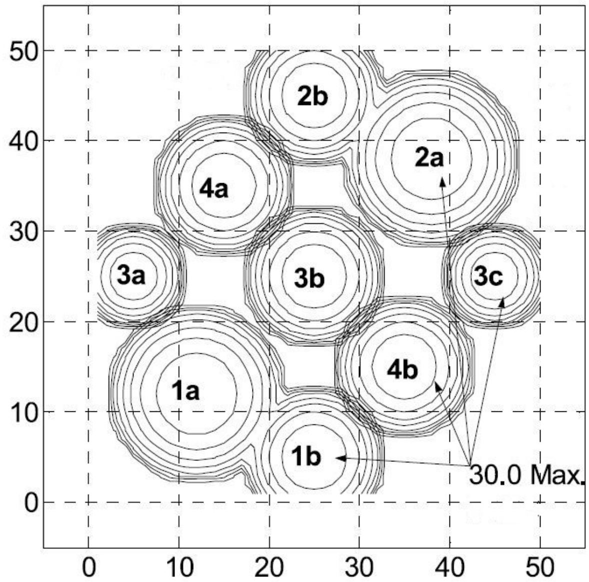

Figure 2.1: Overview of the places of the 9 resource piles in the 50×50 grid community. The sizes represented in this figure are not exactly representative, the figure is meant as overview of the places of the resource piles. The 9 resource piles are divided into 4 different kinds of resources, every resource pile that belongs to the same kind of resource has the same number (but a different letter). The maximum amount of resource in a resource pile is 30.0, and the minimum amount is 1.0. The figure is originally made by Bulleit and Drewek, but adapted for clarification purposes [5].

Over time the area changes. As said in the previous paragraph, the values of the resource piles change due to the fact that civilian agents take away resources. Furthermore, security agents are added and removed from the environment and terrorist attacks take place, these three actions will be explained in more detail later. Also, new civilian agents are added every timestep. This is done in accordance with the Gompertz growth function [20][44]:

$$y(t) = ae^{-e^{bt+1}} \tag{2.2}$$

In which a = asymptotic value, $b = \frac{\text{Maximum slope value}}{\text{asymptotic value}} \cdot \exp(1)$ and t = time. In order to start the civilian agent population with around 500 agents, it has been chosen to translate the graph 500 to the left. Furthermore, an asymptotic number of agents of 1875, 75% of the total area, has been taken, which is in accordance with the statistical findings of Eurostat [18]. In the statistical findings of Eurostat it is stated that “Built-up areas provide a home to almost three quarters (72.4%) of the EU-28’s population”[18]. Moreover, it must be noted that the population growth of the Gompertz function is not representative for real life population growth numbers. This is the case since a population growth factor that is in accordance with for instance the predictions of the United Nations [39] should be at maximum 0.2% per year for the previous and coming years in Europe, as Figure 2.2 suggests. Figure 2.2 presents the previous and expected population growth, according to the United Nations, for all continents in the world. From this figure we can conclude that the population growth should be at maximum 0.2%. Since the size of the population of the model is already not very big, such a small population growth factor would mean that terrorist attacks have a much bigger impact on the society than what would be realistic. The objective of the research is related to the terrorist attacks and not necessarily to the population density and growth factor it is therefore decided that it is in favour of the results to not choose a realistic growth factor. It is however a weakness of the model that the growth factor is not comparable to reality. Since the model should be as realistic as possible in order to create a representative model for reality. Any part of the model that is regarded as unrealistic should be taken into account when analysing the results. The following Gompertz formula is used:

$$y(t) = 1875 \cdot e^{-e^{\left(\frac{2\cdot\exp(1)}{1875}\right)(t+500)+1}} \tag{2.3}$$

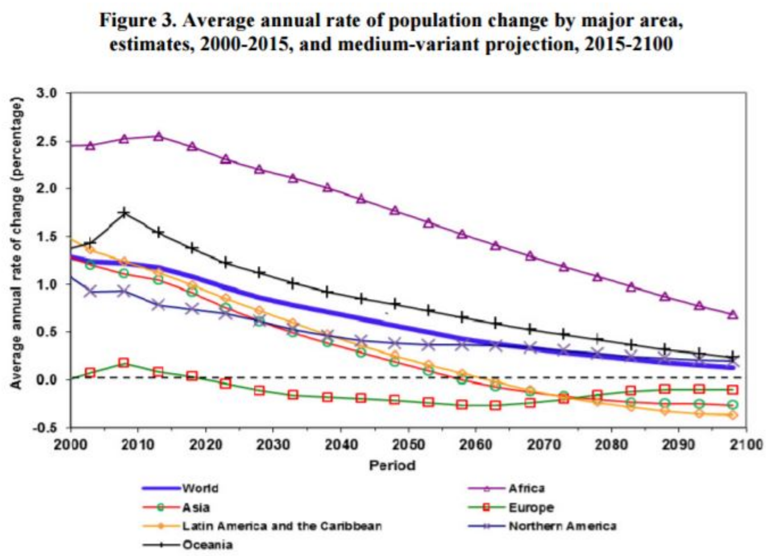

Figure 2.2: Average annual population growth presented by the United Nations, the green line with red circles presents the estimates for the population growth rate in Europe. The figure is originally made by the United Nations [39].

It should also be noted that the model of Bulleit and Drewek did not use the Gompertz function, but had a different implementation. More details about this can be found in paragraph 2.3.

2.2 Agents

Agents are divided into civilian agents and security agents. As Bulleit and Drewek explain, terrorist agents evolve from civilian agents based on their cultural identity. The three different types of agents (civilian, terrorist and security agents) make up the population of the model.

2.2.1 Civilian Agents

Civilian agents are the biggest group of agents present in the model. In order to make the way the civilian agents act and move across the environment as realistic as possible, the model of Bulleit and Drewek uses different kinds of information to define how the civilian agents act: different tags for the behaviour, different resources for the various necessaries, vision and nervousness level to implement the level of fear that is being felt in a society. The initial values of these four different kinds of information are implemented as follows: for the different tags a discrete uniform distribution between 0 and 9 ($U(0, 9)$), for the resources a discrete uniform distribution between 10 and 30 ($U(10, 30)$), for the vision a discrete uniform distribution between 1 and 5 ($U(1, 5)$) and for the nervousness level a discrete uniform distribution between 0 and 1 ($U(0,1)$). The reason why these distributions differ from the article to which Bulleit and Drewek refer is given in paragraph 2.3.

Civilian agents move around the area in the search for new resources. It is, what Bulleit and Drewek call, the metabolism of a resource which indicates on which rate an agent needs to search for new resources. The metabolism is given to each civilian agent at the start of the model or when a new civilian agent is created. Other values that are given at the start of the model or when a new civilian agent is created are the initial resource values for all four different resources present in the model and values for the five different tags. The resource values are the initial amount of resources a civilian gets in order to survive the first time steps. The agent is expected to search for resources him-/herself in order to survive over time. Based on these resource values a Generalised Resource value is calculated for all four different types of resources. This Generalised Resource values indicate how many time steps an agent can survive for each resource, with the assumption that the agent does not collect any more resources. The lowest number of the Generalised Resource values is defined to be the Generalised Wealth. A civilian agent always searches for the resource which is related to the Generalised Wealth, since this resource threatens the life of the agent the most.

As is stated earlier in this chapter the resources are piled up in ‘sugarpiles’, and therefore the area is not necessarily covered with resources everywhere. To solve the problem that agents cannot find new resources, Bulleit and Drewek have given agents memory. Every civilian agent keeps track of the node on which it has seen the biggest value for all four resources. Every agent, as stated before, has been given a vision, and when looking for resources every civilian agent uses this vision to search around to find a new node with a bigger resource value than it already knows. It should be stressed that the agent does not necessarily visit the biggest value it sees within its vision.

The movement of a civilian agent is based on what Bulleit and Drewek call the Adjusted Critical Resource (ACR) value. This value is defined by the following formula:

$$ACR_i = R_i - \Delta R_i \tag{2.4}$$

with

$$\Delta R_i = R_i \cdot n \cdot \left[1 - \exp\left(-\frac{f_i - f_{min}}{f_{max} - f_{min}} \cdot \frac{f_i - f_{min}}{9 \cdot GW}\right)\right], \tag{2.5}$$

in which ACR_i is the Adjusted Critical Resource value for a particular agent at node i, R_i is the resource value at node i, and ΔR_i is the adjusted resource value for an agent at node i. This adjusted resource value is the value that an agent wants to give up in order to lower the fear felt. In the formula for ΔR_i, n is the innate nervousness factor, meaning the factor $0 < n < 1$ which represents how much the agent is affected by the current fear value. f_i is the fear felt at node i, f_{min} is the minimum fear felt by the agent, and f_{max} is the maximum fear felt by the agent. GW is defined to be the value equal to the Generalised Wealth (recall that it is the lowest value of the Generalised Resources). The ACR is calculated for every node in the Moore Neighbourhood, and the agent moves to the node with the largest Adjusted Critical Resource. A graphical representation of the Moore neighbourhood is presented in figure 2.3. In this figure on the left the von Neumann neighbourhood is presented, and on the right the Moore neighbourhood. The blue nodes indicate the neighbourhoods [42] [43]. The von Neumann neighbourhood will be used later in this chapter.

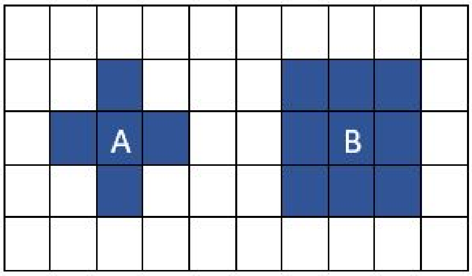

Figure 2.3: Representation of the different neighbourhoods used in the model. Left (Agent A): The blue coloured nodes make up the von Neumann neighbourhood of agent A. Right (Agent B): The blue coloured nodes make up the Moore neighbourhood of agent B.

2.2.2 Evolution Civilian Agent to Terrorist Agent

As is already indicated before, every agent gets assigned five different tag values at the start of time or when the agent is created. These tag values represent five different kinds of characteristics of people and the tags are used for interaction between agents. Every time an agent moves, the moving agent chooses, at random, a node in its von Neumann neighbourhood (see figure 2.3). There are two possibilities, either an agent is on that place, or there is no agent present. If there is no agent, then nothing happens. If there is an agent then the difference S in total tag value is calculated with formula (2.6).

$$S = \sum_{i=1}^{5} |I_{ij} - I_{ik}| \tag{2.6}$$

In this formula I is the value of tag i for person j and k. The probability that agents interact corresponds to the following formula (note that the total possible sum of tag values is equal to $9 \cdot 5 = 45$):

$$1 - \frac{S}{45} \tag{2.7}$$

This means that if the agents have all tags in common, then the probability of interaction is equal to 1, and if the sum S is equal to 45, then the probability is 0.
If the agents interact, then there is an equal 0.20 chance for each tag value to be chosen. For the tag value that is chosen there are two possibilities: either the agent that moves changes its value for that tag to match the other agent, or the agents that moves gets a radical change with probability P_{rc} calculated by the following formula

$$P_{rc} = P_b \cdot \frac{|I_{ij} - I_{ik}|}{9}, \tag{2.8}$$

in which P_b is the base probability, just as Bulleit and Drewek the value 0.03 is used. Whether the tag value becomes a 9 or a 0, is based on the probability of the current tag value of the moving agent. If its current tag value equals 3, then there is a $\frac{3}{9}$ chance that the tag changes to 9, and $\frac{6}{9}$ chance that it changes to 0. We can generally state that the following formula holds for the probability to change to 0:

$$\frac{9 - \text{tag value}}{9} = 1 - \text{probability for tag value to change to nine}$$

After the agents have or have not interacted, there is an isolated probability P_{ic} for a small change in tag value. Bulleit and Drewek have also used $P_{ic} = 0.03$. This small change in tag value means that with equal probability one of the tags is chosen and with equal probability this tag changes by ± 1.

2.2.3 Terrorist Agents

The tag values are also used to determine whether or not civilian agents become terrorist agents. This is done using the following equation

$$\left| \left(\frac{S - 22.5}{22.5} \right)^5 \right|, \tag{2.9}$$

in which S is calculated with equation (2.6). The probability to become a terrorist agent is the biggest near the end values of the cultural identity (which is represented by S), as we can see in figure 2.4.

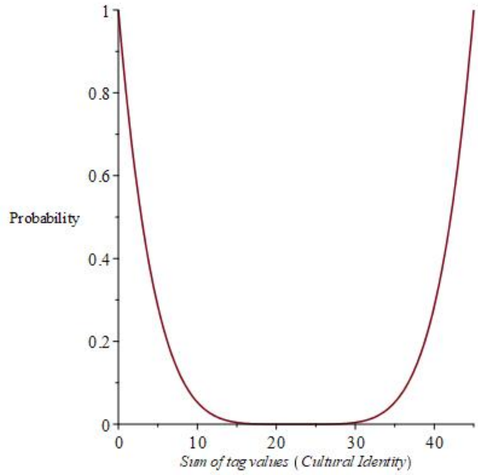

Figure 2.4: Probability that a civilian agent becomes a terrorist agent, according to formula (2.9). The sum of the tag values (x-axis) is calculated by formula (2.6).

The terrorist agent remains an inactive terrorist agent as long as its age is below 18 years, and as long as its Generalized Wealth is below 5.0. Once a terrorist agent is an active agent it remains an active agent as long as its Generalized Wealth stays above 3.0.
An (in)active terrorist agent stops looking for resources, but keeps collecting them, and moves to the nodes with the highest neighbourhood wealth. Terrorist agents keep track of surveillance data, which consists of the summation of the Generalized Wealth of the agents present at time of search and, as Bulleit and Drewek call, the Historical Nodal Wealth. The Historical Nodal Wealth is the moving average of the Generalized Wealth over the past 10 timesteps in the von Neumann neighbourhood of the terrorist agent. Furthermore, an active terrorist agent keeps track of the mean and the standard deviation of the five biggest surveillance values in its surveillance data.
If an active terrorist agent finds a node which meets all of the following requirements then the terrorist agent performs a self-destructing attack. The requirements are:

1. The grid point must have a surveillance data value that is greater than the mean plus one standard deviation;

2. The coefficient of variation of the surveillance data is less than 0.25;

3. There is surveillance data collected from at minimum 5 sites.

As stated before, if the above requirements are met, then the active terrorist agents performs a self-destructing attack in which all resources and agents (and their wealth) are destroyed in the Moore neighbourhood. Note that with this implementation there is at least 1 casualty per attack (the terrorist agent itself).

When an attack has taken place, fear is spread in the area. The amount of fear that is spread is equal to the sum of the Generalized Wealth of all destroyed agents. This fear will diffuse over the area, as Bulleit and Drewek have also implemented it, in correspondence with the model of Epstein and Axtell [17]. Note, to cite Bulleit and Drewek [5, p. 211] and in correspondence with the model of Epstein and Axtell: “new fear scent is not summed on top of existing scent”, meaning that only if the fear scent is bigger than the fear scent already present at the node it wants to diffuse to then it diffuses otherwise the fear scent present at this node is not changed.

2.2.4 Security Agents

The last type of agents are the security agents. Their only task is to arrest terrorist agents and therefore prevent attacks from happening.

Security agents are added or removed from the environment based on the ΔR_i (equation (2.5)). The value for each of the four resources gotten by equation (2.5) is summed up for all non-security agents, which gives a total value that the non-security agents are willing to pay for security per resource type. This value is then divided by the average metabolism for each resource for all non-security agents. The lowest of these four values is the amount of security agents that are present at the next timestep. If there are currently more security agents present, then security agents are removed with a higher probability for security agents that are situated in areas of low wealth than for agents situated in areas of higher wealth. And if there are currently less security agents present, then security agents are added with a higher probability that security agents are added in areas of high wealth than in areas of lower wealth.

Security agents arrest terrorist agents by examining, at random, one of the nodes in their Moore neighbourhood on which a non-security agent is placed. If the examined agent is a civilian agent, then it is released. If the examined agent is a terrorist agent (active or inactive) then with a probability calculated with equation (2.10) the terrorist agent is arrested.

$$\left| \left(\frac{S - 22.5}{22.5} \right)^3 \right| \tag{2.10}$$

Similar to the probability that an civilian agent becomes a terrorist agent, the probability to be arrested as terrorist agent is higher if the terrorist agent has more extreme tag values, as we can derive from figure 2.5

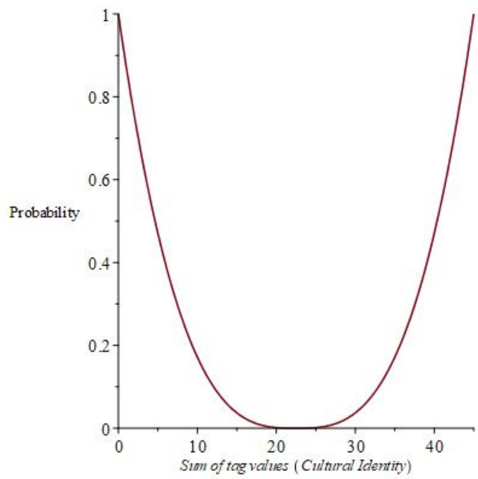

Figure 2.5: Probability that a terrorist agent is arrested by a security agent if it is examined, according to formula (2.10). The sum of the tag values (x-axis) is calculated by formula (2.6).

Security agents move to the node that has the most non-security agents in their Moore neighbourhood.

2.3 Differences implemented and original model

With respect to the model of Bulleit and Drewek, explained in *Agent-Based Simulation for Human-Induced Hazard Analysis* (2011) [5], the implementation used for this research has some differences, motivation for these differences will be given per change that has been made.

First of all some small details of the model of Bulleit and Drewek have not been implemented. The details that have not been implemented are:

- In the model of Bulleit and Drewek, once an attack has taken place the resource values do not grow back for two years. Since the code became very slow when this was implemented, it has been decided to leave it out. With the implemented model the simulations are already very time-consuming and to be able to simulate at least a decent number of times we should take care of the time that each simulation takes. The expected consequence is that the destroyed resource value per attack is higher. This is expected since the resources grow back earlier, and therefore the total resource value present in the area is higher, thus on average an attack is expected to destroy more resources. For future models it is suggested to do more research on how to make the used code (which can be found in Appendix B) faster.
- Opposed to Bulleit and Drewek, no incubation period was used because a different implementation of the population growth is used. Bulleit and Drewek used the incubation period in order to get a stable number of agents in the model. Since we used a different implementation for the growth factor of the number of agents (which will be explained and motivated later in this section) there is no need to get a stable population, since it will already be stable. Therefore, it is believed that no incubation period is necessary.

Furthermore, some details have been implemented differently, these are:

- In the model of Bulleit and Drewek, the number of non-security agents present in the model was sustained by agents that die and agents that are born. The dying of agents once their generalised wealth is below 1.0 is implemented in a similar way as Bulleit and Drewek did. However the way civilians are born is implemented differently. As can be read in paragraph 2.2.1 civilian agents are now 'born' based on the Gompertz function, whilst Bulleit and Drewek based the creation of new civilian agents on the reproduction between male and female agents. This implementation has been tried, but resulted in a drastic decrease in the total population as can be seen in figure 2.6. Starting with around 500 agents, resulted in several simulations that the total number of agents decreased very fast to around 50-100 agents. Also trying to use an incubation period as Bulleit and Drewek used did not result in a stable number of agents. Both intuitively and based on figure 2.2 we would conclude that a fast decreasing number of agents in the model is not representative for society. It has therefore been decided to use the Gompertz function. The Gompertz function, as has been said, does also not represent the growth factor of the population of society but it does represent a steady growth.

- In the introduction of this chapter it has been said that two different articles have been used for the implementation of the model by Bulleit and Drewek. The implementation is mainly based on the article *Agent-Based Simulation for Human-Induced Hazard Analysis* (2011) [5], but the initial distributions of the information for civilian agents are based on the ideas behind the distributions explained in *Simulating Initial Conditions in Agent-Based Modeling* (2005) [6]. It has been chosen to use different distributions than explained in this last article, with the reason that the article uses also other different assumptions as the article we use for this model (The 2011 version of the model of Bulleit and Drewek). For example, the resource piles have a minimum value of 1.0 and a maximum value of 54.0, whilst we use a maximum of 30.0. Since the assumptions differ, the distributions have been changed as well in order to keep a similar model. In order to keep the model as similar as possible, the distributions explained in the 2005 article by Bulleit and Drewek have been used as reference [6].

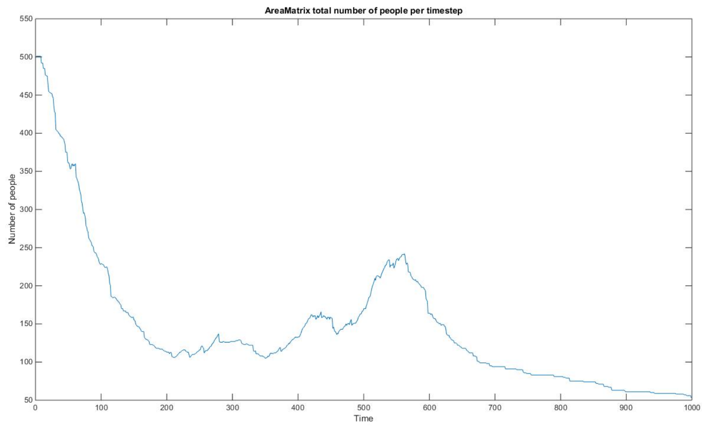

Figure 2.6: Simulation with an initial amount of 500 civilian agents, running for 1000 timesteps. The graph shows the amount of agents present in the model at every timestep.

There are also a few implemented details that might make the model a bit less realistic. Of course it is a consideration whether or not it is in favour of the model to change this. Making the model more detailed might make the model very realistic for certain situations, but might also rule out other situations. Another reason why certain details might not be included is that we want the the model to represent reality whilst being as basic as possible, every detail of a model can be thought of as not realistic in certain situations. The following details are considered to be less representative for a community:

- The uniform distributions of each of the tag variables. All tag variables combined will represent that it is much more unlikely to have an extreme view than an average view. However, intuitively it is also less likely to have an extreme view on a group of characteristics (one of the tags) instead of a equal probability to have such a view. The equal probability is implemented in the model with the uniform distribution. This implementation might have as consequence that it is more likely to have extreme characteristics in the model than in society. If this consequence is true than it will most likely cause the number of terrorist agents to be higher than in society.

- The agents are moving only one node to any side at the time. However since we use discrete time intervals it might be more representative to give the agents some kind of possibility to jump to nodes that are further away.

2.4 Conclusion

We have now seen the implementation of an Agent-Based model for terrorist activity partially based on the article of Bulleit and Drewek. The model simulates a normal society and is based on three agents: civilian, terrorist and security agents. In the last paragraph the model has been put to question and some details that changed compared to the implementation of Bulleit and Drewek have been explained. These details include for instance the creation of new civilian agents and the distributions of initial information for the agents. In the next chapter the data that is stored from several simulations of this model will be analysed.

Chapter 3

Statistical analysis

The model described in Chapter 2 (see the code in Appendix B, page 65) has been implemented in Matlab [37] and simulated 11 times for analysis of the model. Each of the simulations took the computers used on average 4.5 hours to complete. Each simulation used, besides the conditions mentioned in chapter 2, also the following initial conditions:

- The area is a 50$\times$ 50 grid,
- Initial amount of civilian agents equal 500,
- Total number of timesteps equal 1030,
- 5 non-security agent tag variables,
- 9 resource piles, with 4 different types of resources. The highest points of the sugar piles placed in accordance with figure 2.1.

First the general results will be shown and explained, these include the total number of people, and a histogram of the percentage security agents present at every timestep. After that we will check whether or not the results (number of casualties and amount of damage for the attack) of the simulations follow similar distributions. A short explanation of the used distributions can be found in Appendix A (page 63). Thereafter, the size of attacks will be analysed by the means of several tests. Last but not least, the waiting time and number of attacks per time period are discussed.

It should be stressed that for statistical analysis 11 simulations is not a substantial amount. The simulations are very time-consuming (as stated before: 4.5 hours on average per simulation) and therefore we have unfortunately not been able to do more simulations.

Each of the 11 simulations were used for the analysis of the model, however only 1 simulation is used to provide the figures, unless stated differently. When we talk about the ‘combined dataset of the eleven simulations’ or simply the ‘combined dataset’, it is meant that the data of the eleven simulations is joined together into one big list. For the statistical analysis we have used explanations from Rice [30].

3.1 General overview results

Bulleit and Drewek indicated that each timestep is representative for one week [5]. In order to justify this for the implemented model, we will compare the average number of attacks per timestep with the average number of attacks per week in the Global Terrorism Database. For this comparison we will use a simple z-test, which compares the average of the combined list of casualties per timestep with the average casualties per week in the GTD. The test will reject the null hypothesis: The means of the two datasets are equal, in favour of the alternative hypothesis: The means of the two datasets are not equal, if the means of the two datasets are significantly different. A significance level of 0.05 is used. The z-test returns a p-value equal to $4.5850 \cdot 10^{-20}$, and therefore the null hypothesis is rejected with the significance level of 0.05 used. Meaning that the mean of the attacks per timestep of the eleven simulations is considered not be equal to the mean of the Global Terrorism Database per week with this significance level.

In order to find a good corresponding timestep, several combinations have been tried. Such as 4 timesteps equal 1 month, and 2 timesteps equal 3 weeks. With the z-test the combination 10 timesteps equal 3 months has been successful. With the same null hypothesis, alternative hypothesis and rejection level the z-test returned a p-value of 0.2132, meaning that the null hypothesis is not rejected and that the means of the two datasets are considered not to be significantly different.
In the remaining analysis of this chapter we will use that 10 timesteps of the simulations equal 3 months in reality (according to the comparison with the Global Terrorism Database).

The amount of agents present was modelled with the Gompertz function (as is explained in section 2.1). This caused the population density to be equal in all simulations, and the growth of the population is presented in figure 3.1. As expected the growth factor of the number of people present is not representing a realistic number, however we do see the expected positive growth factor. Furthermore, every simulation has the same growth pattern, this is of course also a weakness of the model. It would be much more realistic to see some differences in population growth between the various simulations. As is stated before these weaknesses might be influencing the results. We should therefore be cautious when interpreting the results, and keep these weaknesses in mind.

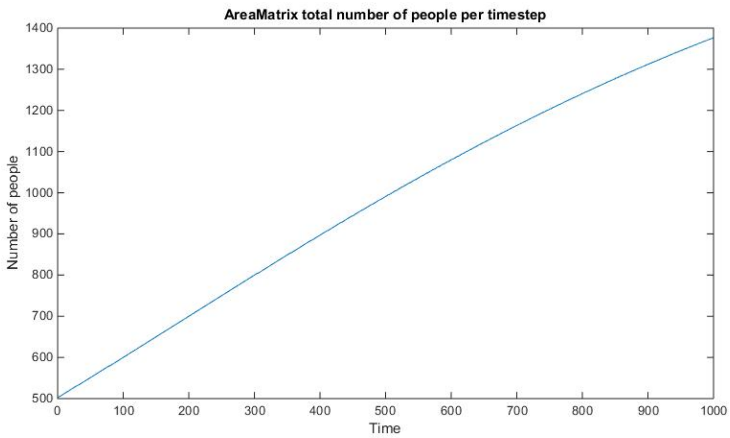

Figure 3.1: Number of agents per timestep for the simulation time of 1030 timesteps. Initially there are 500 civilian agents, and the growth factor is determined by the Gompertz function. The growth factor from the simulations is not a representative growth factor for society.

There is a clear difference between the distribution of agents at the beginning of the simulation and at the end of the simulation. As we can see in figure 3.2, the first 150 timesteps (left picture) result in a clear clustered distribution of agents around the centre resource pile (please use Figure 2.1 for reference of resource piles), whilst the last 150 timesteps (right picture) show a more spread out distribution. The dark red colour represents relatively many agents passed that node, a white colour represents almost no agents that passed the node. This clear difference in distribution of the agents might be caused by the various attacks that have happened, and that the civilian agents (representing the biggest group of the society) fear the original resource wealthy nodes. It might also be caused due to the implementation that non-security agents take away resources, and that therefore it is more efficient for these agents to be spread throughout the area in order to obtain more resources.

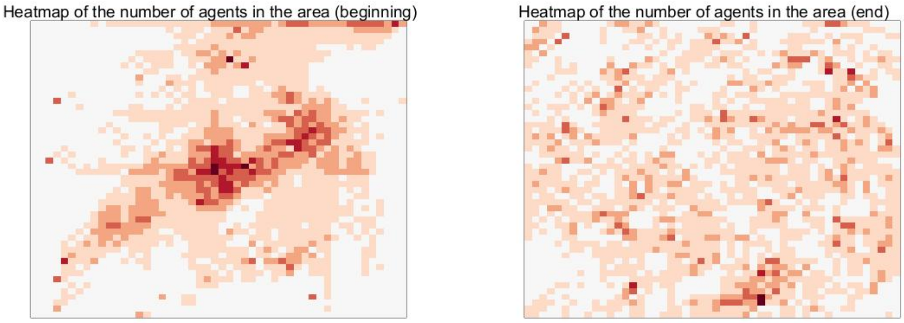

Figure 3.2: Distribution of number of agents in the simulated community of one of the simulations. Left: the distribution of the first 150 timesteps of the simulation. Right: the distribution in the last 150 timesteps of the simulation (timestep 880-1030). The dark red colour represents relatively many agents passed that node, a white colour represents almost no agents that passed the node.

The percentage security agents present in the model over time is represented in figure 3.3. As we can see in most cases the number of security agents present in the model increases as bigger attacks have taken place at the same time. This is caused due to the fear an attack spreads throughout the area. We can see that there are a lot more terrorist agents after the attacks that happen in the beginning of time than at the end, this is caused by the attacks at the end of time being smaller than at the beginning, which can be seen in figure 3.4. If attacks are smaller, then also less fear is created and therefore less security agents are needed according to the model.

As we can see in figure 3.3 the percentage of security agents is between 0% and 10%, it is good to note that this matches with the percentage shown in figures on page 20 of [6] by Bulleit and Drewek.

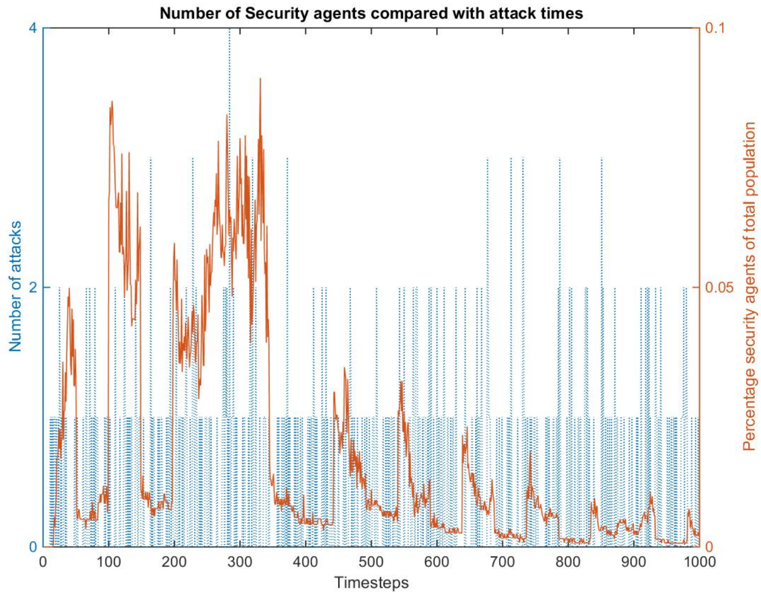

Figure 3.3: Percentage of security agents present in the model (red line), the peaks in the graph correspond to the attacks that produced relatively a lot of fear by the non-security agents in some timesteps earlier. The blue bars show the amount of attacks that have taken place on each timestep.

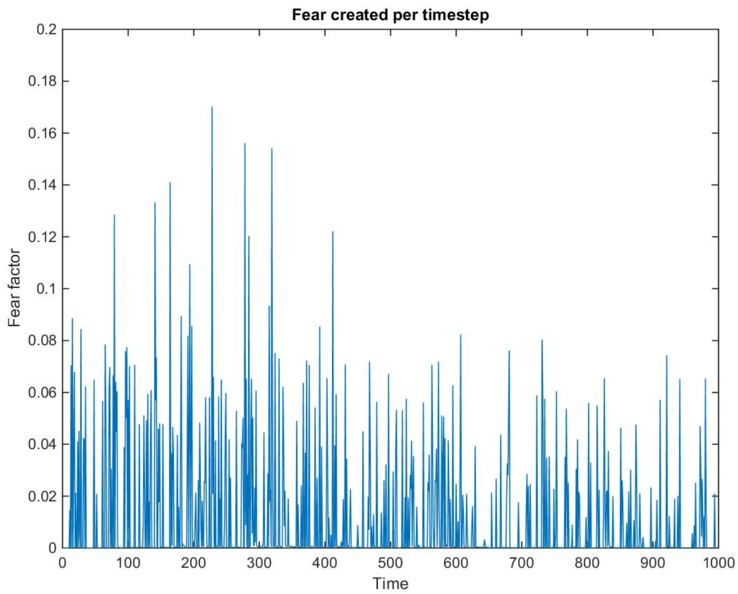

Figure 3.4: Fear factor that is created by the attacks per timestep. As can be seen the fear factor reduces over time, meaning that the impact the attacks have on agents also reduces.

This makes the general overview look quite positive, let us now get into more detail with a comparison of the distributions and by using different statistical tests.

3.2 Size of attacks

The size of the attacks can be defined in two ways: first of all in the number of casualties created by the attack and secondly in terms of the amount of damage that is done. Both these definitions for size of an attack will be analysed by comparing the average of the combined simulations with the average of the Global Terrorism Database, by comparing the averages of each of the simulations with the average of the Global Terrorism Database and by comparing the distribution of the size of attack of the model with distribution presented by Bulleit and Drewek and by comparing it with the distribution derived from the Global Terrorism Database.

3.2.1 Casualties

Casualties are in the model defined as the total number of agents that are wounded and have died. We will therefore compare the amount of casualties in the model with the total number of people killed and wounded in attacks according to the Global Terrorism Database. It has also been chosen to not include the terrorist attacks without any casualties, since these kind of attacks are also not included in the model. An attack in the model is implemented in such a way, that there will be at least 1 victim (the terrorist agent). Bulleit and Drewek do not give a distribution that relates to the number of casualties according to their model.

Comparing mean and variance

The Global Terrorism Database has a mean of 13.06 casualties per terrorist attack that had at least 1 casualty. Taking into account all eleven simulations as a combined dataset then the mean number of casualties per attack equals 14.08. As we can notice the mean number of casualties is only a little bit higher in the simulation than it is in reality according to the Global Terrorism Database, from this we would expect that the means of both datasets might be considered not to be different. Our expectations are satisfied with a simple z-test, this test does not reject the null hypothesis. We have used as null hypothesis that the mean number of casualties in the eleven simulations combined equals the mean number of casualties in the Global Terrorism Database, and as alternative hypothesis that these means are not equal. The significance level is again 5%. The z-test gave a p-value of 0.1730 and that indicated that the null hypothesis should not be rejected. It also means that the mean number of casualties of the two datasets are considered not to be different. The number of a little less than 15 casualties per attack is also supported by findings of Frykberg and Tepas [19], they found an average of 15.3 casualties per attack by analysing 220 attacks worldwide.

It should be noted that if the number of wounded and killed people is not combined in the dataset of the GTD, then the mean of the number of killed people according to the GTD would be equal to 3.01. This is a much lower number, and therefore expected to be considered not equal to the mean of the combined dataset of the simulations of the model. Using the z-test with the same null hypothesis, alternative hypothesis and significance level as in the paragraph above, we would in this case indeed have to conclude that the mean number of the killed people per attack according to the GTD is significantly different from the mean number of killed people in the model. Therefore, if the number of wounded and number of killed people in the GTD is not combined, it is suggested to change the model in such a way that the means do match.

When looking at the variances, the combined dataset of the eleven simulations shows a variance of 87.5622 whilst the variance of the Global Terrorism Database equals 2128. We would therefore expect the variances to be considered different. Testing with a chi-Squared variance test whether or not the variances for the eleven simulations combined and the data from the Global Terrorism Database are the same gives, as expected, a negative result. Using a significance level of 0.05, the null hypothesis that the variances of the two datasets are equal is rejected in favour of the alternative hypothesis that the variances are different since the p-value that the test returns equals 0.

Multiple comparison test

The simulated data for all eleven simulations combined does compare with the Global Terrorism Database considering the mean of the number of casualties per time period. We can also test whether each of the simulations does compare with the Global Terrorism Database as well as whether they compare with each other individually. For this analysis we will make use of the Bonferroni correction and the multiple comparison test. We will use this correction since we will test multiple hypotheses, therefore the chance that a rare event happens increases. An increase in the possibility that a rare event happens will cause it to be more likely that we have to reject the null hypothesis. Rejecting the null hypothesis incorrectly is also called 'making a type-1 error' (accepting the null hypothesis whilst the null hypothesis should be rejected is called a 'type-2 error'). The Bonferroni correction compensates this type-1 error partly.

Figure 3.5 shows the outcome of the multiple comparison test using the Bonferroni correction. The multiple comparison test uses for every pair of datasets as null hypothesis that the means are the same and as alternative hypothesis that the means are not equal. The means are considered not to be equal by the multiple comparison test if the bars that are present in Figure 3.5 (for the pair of datasets considered) do not overlap. As we can see all means are considered not to be different of each other, since all pairs of datasets have overlapping bars. We can therefore also see in Figure 3.5 that we do not reject the null hypothesis when we only look at the eleven simulations. The eleven simulations are considered to not have a different mean for the number of casualties. This means that the model is quite consistent in the results it gives.

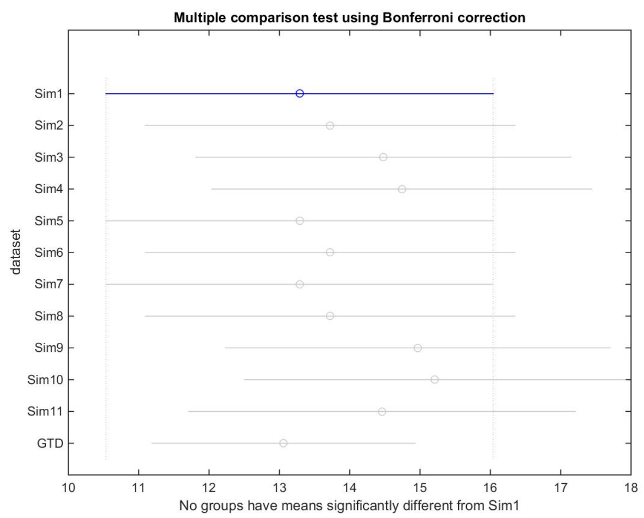

Figure 3.5: Multiple comparison test with all eleven simulations separate and the dataset of casualties from the Global Terrorism Database using the Bonferroni correction. The multiple comparison test tests which datasets are considered to have equal means and which datasets are considered to have different means.

Comparing distributions

Comparing the underlying distributions from both the eleven simulations and the GTD, let us start with comparing the data with a Quantile-Quantile plot, as is shown in Figure 3.6. From this figure we have the expectation that the GTD and the dataset of combined casualties from the simulations do not follow the same distribution, since the data points are not in a straight line.

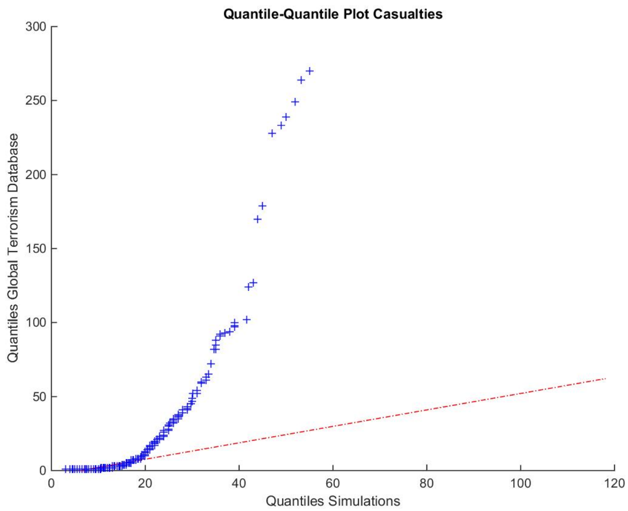

Figure 3.6: Quantile-Quantile plot of the number of casualties per timestep (for the simulations) and per 3 months (for the Global Terrorism Database). The straight line through the minority of the points shows that the underlying distribution of both data sets are expected to be different.

We can also compare the histograms of both data sets, this is shown in Figure 3.7. Also from this figure we see quite some difference. The dataset of the GTD shows many more attacks that had a low number of casualties compared to the dataset of the simulations. Therefore, also from this figure we would expect that the two datasets do not follow the same distribution.

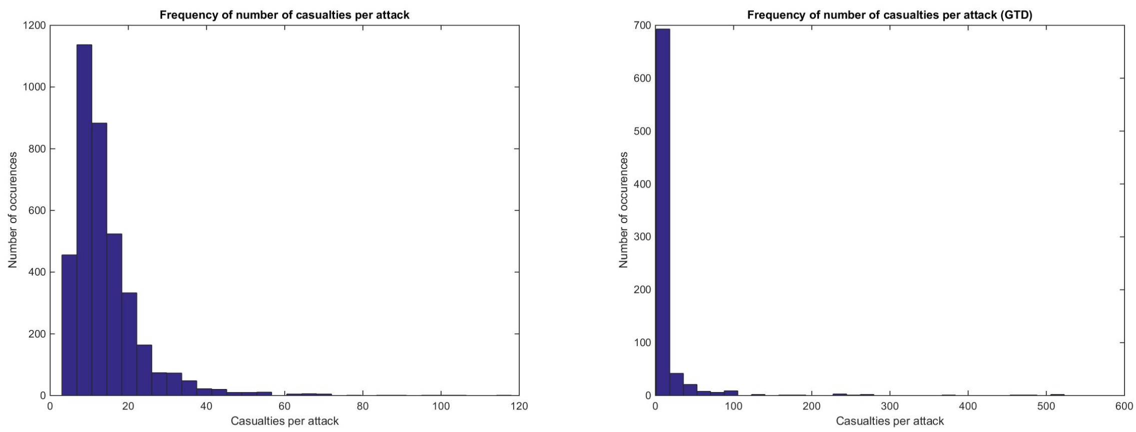

Figure 3.7: Histogram of the frequency of number of casualties for the eleven simulations combined (left) and the Global Terrorism Database (right).

A two-sample Kolmogorov-Smirnov test is used to determine whether or not the underlying distributions of the Global Terrorism Database and the eleven simulations differ. The test uses as null hypothesis that the distributions are similar and as alternative hypothesis that states that the underlying distributions differ significantly. A significance level of 0.05 is used. A p-value of $1.4866 \cdot 10^{-276}$ is returned, and therefore with the significance level of 0.05 the null hypothesis is rejected. This means that the underlying distributions of the two samples are considered to be different based on this information.

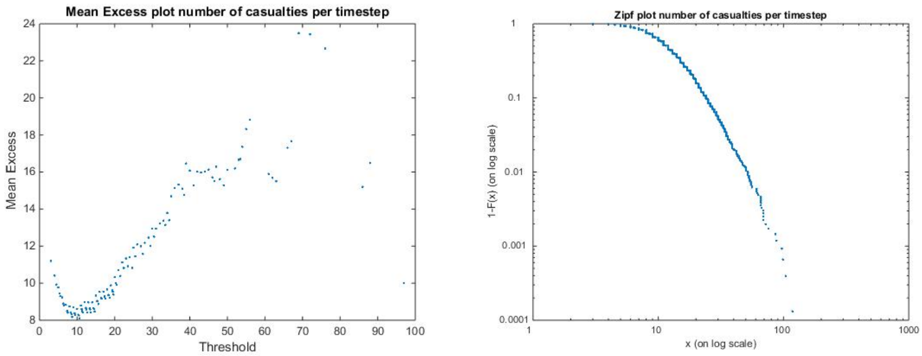

Figure 3.8: Left: Mean Excess plot of the number of casualties per timestep. As can be seen after a threshold of 10 we can see some starting Paretianity by the linearity of the data. Right: Log-log plot (or Zipf plot) of the empirical survival function. This plot shows that for more than 10 casualties per timestep the data shows some Paretianity.

Scharpf, Schneider, Nöh, and Clauset [33], Clauset and Woodard [13] mention in their research that the number of casualties in war conflicts are Pareto distributed. To test whether or not the model follows such an distribution we use a Mean Excess plot of the number of casualties per timestep and a Log-Log plot (or Zipf plot) of the empirical survival function, which are shown in Figure 3.8. Instead of using a Quantile-Quantile plot we are using these types of plots since a Quantile-Quantile plot is less reliable for visually analysing whether or not data is Pareto distributed, the combination of a Zipf plot and a Mean Excess plot is much more reliable [12]. As we can see from both plots in Figure 3.8 we are likely to have Paretian tails. We derive this from both plots since the Zipf plot (right plot in Figure 3.8) shows linearity of the data for $x > 10$ (where x are the ordered data). Also the clearly upward trend for more than 10 casualties per timestep in the Mean Excess plot (left plot in Figure 3.8) gives us a sign of some Paretianity. We are, however, not satisfied with the tail parameter of around 10, since we expect a parameter of 2.91, according to Clauset and Woodard [13]. Therefore, either more replications are needed in order to do a better analysis, or we should change the model. For further analysis it is good to note that Paretianity is found.

A chi-squared goodness-of-fit test is used to test whether the dataset of the simulations combined follows a Pareto distribution. The test uses as null hypothesis that the dataset follows a Pareto distribution and as alternative hypothesis that states that the distributions differ significantly. With a p-value of 0 we will reject the null hypothesis. A significance level of 0.05 is used. Various parameters for the Pareto distribution have been tried, but none resulted in a better p-value. Based on the chi-square goodness-of-fit test We will therefore, consider the distributions to be significantly different.

It should be noted that Virkar and Clauset [40] mention that with statistical evidence it is hard to prove whether or not empirical data follows a Pareto Distribution due to large fluctuations in the tail. From the Zipf plot and the Mean Excess plot in Figure 3.8 we can clearly see that the tail of the data shows some Paretianity. It might, however, still be one of the weaknesses of the model that the number of casualties is not distributed as the GTD and that the number of casualties does not perfectly follow a Pareto distribution if we compare our data to reality. From the Quantile-Quantile plot in Figure 3.6 we can conclude that the data of the simulations is believed to have too few attacks that have many casualties in comparison with the GTD.

3.2.2 Damage

Besides the attack size in terms of casualties, we can also analyse the attack size in terms of damage that is done. Bulleit and Drewek state that the "The attack magnitudes fit a log-normal distribution"[5, p.214], so we will check whether or not the implemented model shows the same result. The Global Terrorism Database does not have a good summary of the total damage done by terrorist attacks.

Multiple comparison test

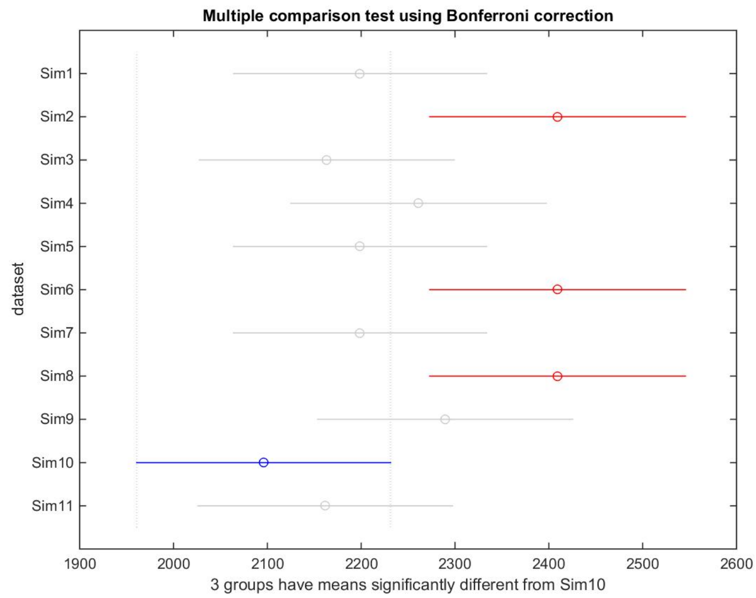

Figure 3.9: Multiple comparison test with all eleven simulations separate using the Bonferroni correction. The multiple comparison test tests which datasets are considered to have equal means and which datasets are considered to have different means. The dataset belonging to simulation 10 (Sim10) is considered to have a different mean compared to the datasets of simulations 2, 6 and 8. This is indicated by the blue and red colours.

Using the multiple comparison test with the Bonferroni correction we can check whether the means of the eleven simulations are considered to be equal. Figure 3.9 shows the outcome of the multiple comparison test using the Bonferroni correction. Please recall that the means are considered not to be equal by the multiple comparison test if the bars that are present in figure 3.9 (for the pair of datasets considered) do not overlap. If for the pair of datasets considered the bars do overlap, than the means are equal according to the multiple comparison test. As can be noticed in figure 3.9, the outcome of the multiple comparison test for simulation 10 (sim10) is coloured blue whilst simulations 2, 6 and 8 are coloured red (sim2, sim6 and sim8). This means that the dataset belonging to simulation 10 (Sim10) has a different mean compared to the datasets of simulations 2, 6 and 8. Checking whether or not any other pairs have overlapping bars shows us that any other combination is regarded as having the same mean. This means that if we would leave out simulation 10 we would not reject the null hypothesis that any simulation does not equal the mean of another simulation regarding the damage done by the attack if simulation 10 is not taken into account. Thus the model is not perfectly, but still quite consistent in the results it gives.

Comparing distributions

As is said earlier, Bulleit and Drewek found that the attack magnitudes (damage caused by terrorist attacks) follow a log-normal distribution, they have also found evidence in other research that also suggested the attack magnitudes to be log-normally distributed. This evidence is for example given by Cederman [9].

The histogram of the resource values destroyed per attack is shown in figure 3.10. This figure shows that the data is expected to follow a log-normal or exponential distribution.

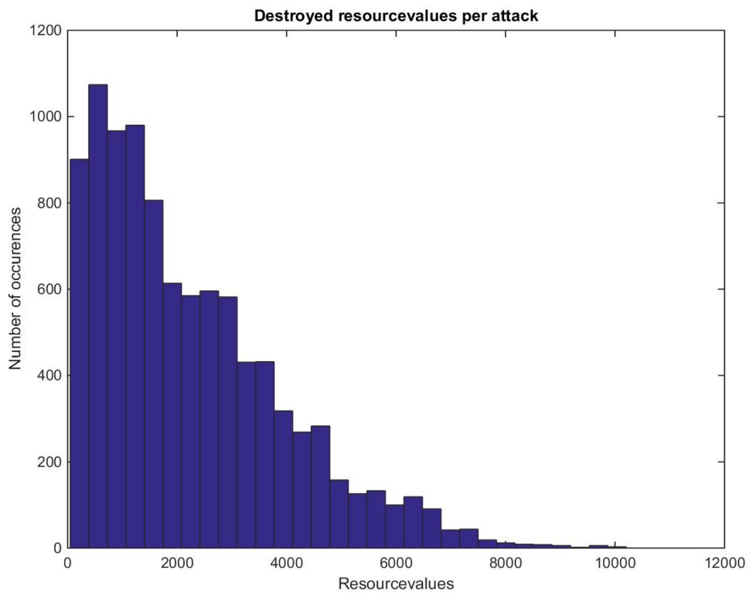

Figure 3.10: Histogram of the destroyed resource values per attack for the eleven simulations combined.

The Quantile-Quantile plots in figure 3.11 shows on the other hand that neither an exponential nor a log-normal distribution is expected. We can conclude this from figure 3.11 since both Quantile-Quantile plots show that the data is not presented on a straight line.

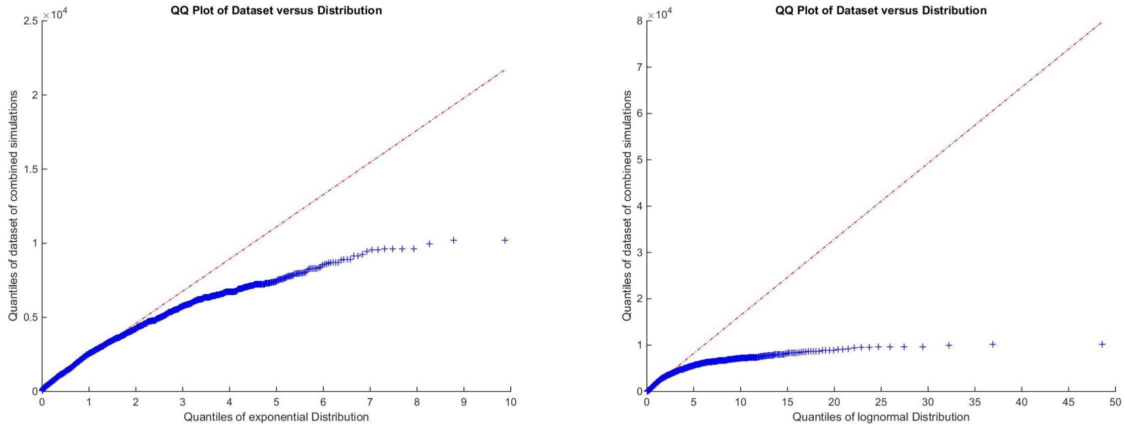

Figure 3.11: Quantile-Quantile Plot of the destroyed resource values per attack against the theoretical exponential (left) and log-normal (right) distributions.

From the Kolmogorov-Smirnov test, the Anderson-Darling test and the Lilliefors test we can all conclude that we should reject the null hypothesis that the destroyed resource values per attack are log-normally distributed in favour of the alternative hypothesis that the destroyed resource values per attack do not follow a log-normal distribution with a significance level of 0.05. The Kolmogorov-Smirnov test returns a p-value of $2.0054 \cdot 10^{-42}$, the Anderson-Darling test a p-value of $5.0000 \cdot 10^{-04}$ and the Lilliefors test a p-value of $1.0000 \cdot 10^{-03}$. It also holds that the tests reject the null hypothesis that the data follows an exponential distribution in favour of the alternative hypothesis that the data does not follow such a distribution. The p-values are: $1.2225 \cdot 10^{-53}$ (Kolmogorov-Smirnov test), $5.0000 \cdot 10^{-04}$ (Anderson-Darling test) and $1.0000 \cdot 10^{-03}$ (Lilliefors).

In figure 3.11 we do, however, see that the exponential distribution is fitting better than the log-normal distribution. We will therefore try a generalisation of the exponential distribution: the Weibull distribution. As we can see in figure 3.12 the data is following the Weibull distribution even better. However the Weibull distribution is still not expected to be the right distribution due to the tail of the data being different than the theoretical values.

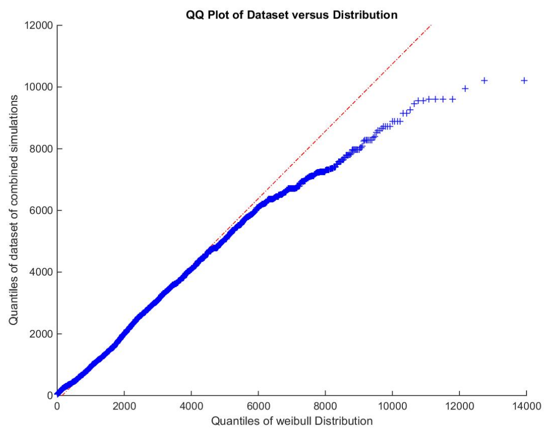

Figure 3.12: Quantile-Quantile Plot of the destroyed resource values per attack against the theoretical Weibull distribution.

Using the Kolmogorov-Smirnov and the Anderson-Darling tests, we again reject the null hypothesis that the data is Weibull distributed in favour of the alternative hypothesis that the data does not follow such a distribution. A significance level of 0.05 is used. The Kolmogorov-Smirnov test returned a p-value of $6.0089 \cdot 10^{-09}$ and the Anderson-Darling test a p-value of $5.0000 \cdot 10^{-04}$.

It is therefore unknown which distribution suits the data, though the Weibull distribution looks like it is best fitting. As is stated earlier, literature shows that a log-normal distribution is expected. More research would be needed in order to get a better understanding of the distribution of destroyed resource values.

3.3 Waiting time

Bulleit and Drewek found that the time between the attacks (also called the waiting time) is exponentially distributed. With the histogram in figure 3.13, we would expect the same for the combined dataset of all eleven simulations.

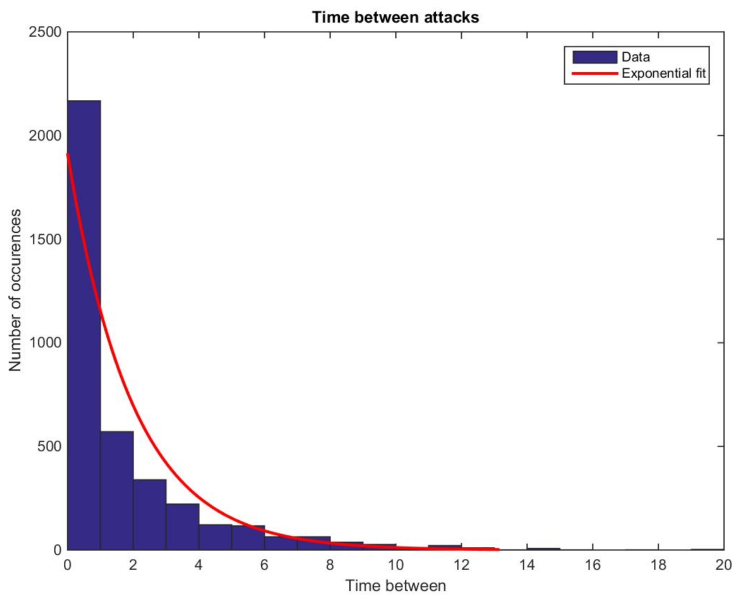

Figure 3.13: Histogram of the waiting time between attacks for the dataset of the data of the eleven simulations combined. It suggests an exponential distribution. The red line represents the fitted exponential distribution.

Also the Quantile-Quantile plot, as can be found in figure 3.14, of a fitted exponential distribution versus the combined dataset of Times between the attacks shows that we expect the data to be exponentially distributed.

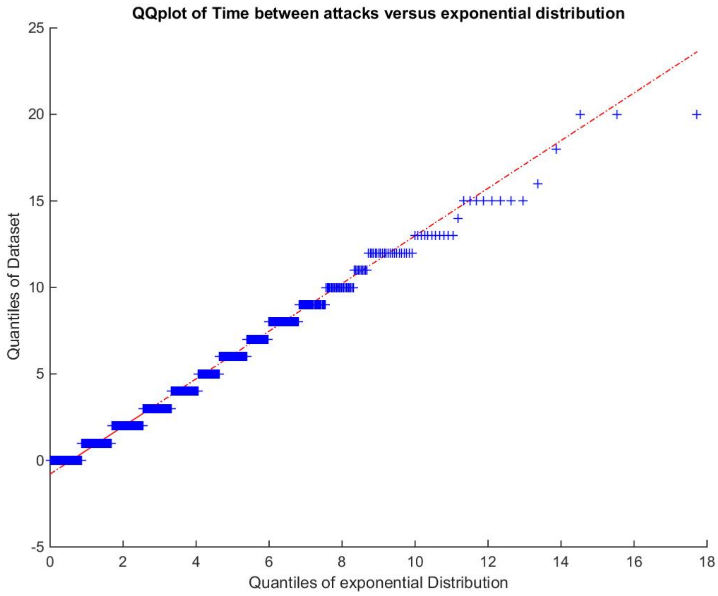

Figure 3.14: Quantile-Quantile plot of the combined dataset of the times between attacks versus the theoretical exponential distribution.

However, a chi-squared goodness-of-fit test (p-value of $8.2272 \cdot 10^{-50}$) and a Kolmogorov-Smirnov test (p-value of 0) show both that we have to reject the null hypothesis that the data is exponentially distributed in favour of the alternative hypothesis that it is not distributed in such a way. A significance level of 0.05 has been used.

We should, however, note that both these tests are very conservative. In figure 3.13 the fitted exponential distribution has been added to the Histogram. As we can see this looks, as stated before, very much like the exponential distribution. As does the Quantile-Quantile plot. We should note that both tests are more likely to reject the null hypothesis since we are comparing a discrete dataset with a continuous distribution. This difference is also the cause of the 'stair' behaviour in the Quantile-Quantile plot in figure 3.14.

3.4 Number of attacks

Figure 3.15 shows the number of attacks over 1000 timesteps for one of the simulations. As can visually be noted, the number of attacks are expected to be more or less evenly distributed over time.

However, the number of attacks are considered to not be a homogeneous Poisson process by the chi-squared goodness-of-fit test. We can conclude this since the null hypothesis is rejected in favour of the alternative hypothesis. The hypothesis tested (null hypothesis) is whether each of the bins has the same probability that the number of attacks measured in the bin occurs against the alternative hypothesis that at least one of the probabilities is different. The p-value returned by this test is $5.006 \cdot 10^{-12}$. The number of attacks is on average not constant over time if we split time into non-overlapping bins of 80 timesteps. A significance level of 0.05 is used. Other sized bins also rejected the null hypothesis.

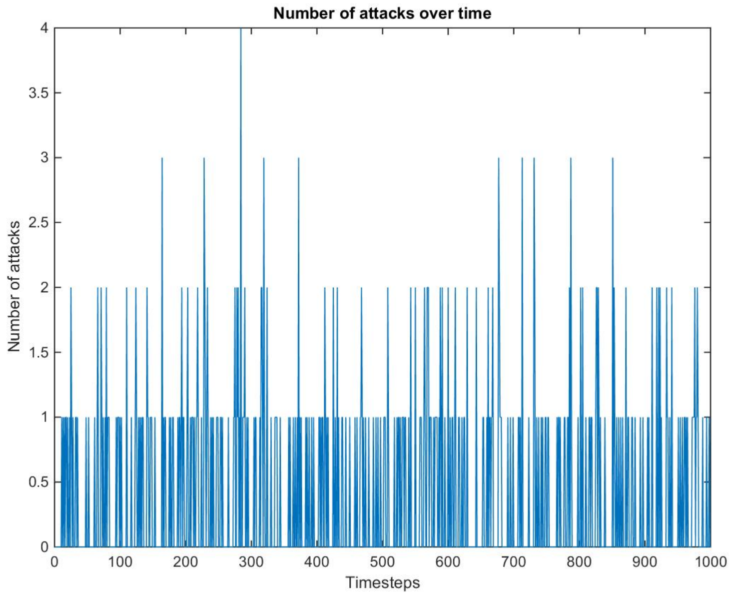

Figure 3.15: Number of attacks over 1000 timesteps for one of the simulations.

Also the multinomial test returns that the number of attacks are considered to not be a homogeneous Poisson process. The null hypothesis tests that the parameters of a multinomial distribution are all equal for each bin. The alternative hypothesis states that at least one of these parameters is considered to be different. Since the total number of observations (more than 11000) is very large, we have used the option 'Monte Carlo approach' in order to keep a reasonable time this test takes. The p-value returned was equal to 0, meaning that we will reject the null hypothesis in favour of the alternative hypothesis with a significance level of 0.05 and the same bin size as with the chi-squared goodness-of-fit test above. It is therefore also by the multinomial test believed that the number of attacks are not a homogeneous Poisson process. This means that the number of attacks varies over time, on average. Which might imply that terrorist attacks influence each other, meaning that if an attack has taken place it is less likely or more likely that another attack takes place. Or it might for example imply that terrorist attacks are clustered, meaning that terrorist agents are more likely to perform their attacks around the same time. We should also take into account that compared to the model of Bulleit and Drewek no incubation time has been used, which could also be a cause that the number of attacks varies over time. Further analysis is needed in order to get a better understanding of the number of attacks over time.

3.5 Conclusion

Based on the statistical analyse done in this chapter, we can conclude that the implemented Agent-Based model which is based on the model described by Bulleit and Drewek does not really represent theoretical findings. The timesteps in the model are representing just a little bit more than a week (10 timesteps equal 3 months) and the mean number of casualties per attack is considered not to be different compared to the Global Terrorism Database. However, whilst quantile-quantile plots and histograms show some relation with the theoretical expectations, none of the underlying distributions are accepted to not be different by several tests. It is one the other hand found that the number of casualties per timestep show some Paretianity in the tails. Furthermore, it was found that the attacks are not a Poisson process.

The comparisons are summarised in the table below. In this table is stated if a finding is similar or not similar to the findings stated by Bulleit and Drewek (named 'Model' in the table), findings retrieved from the GTD or findings by other theoretical research. A '-' represents that no information is found in the report of Bulleit and Drewek, in the GTD or no other theoretical research is used for analysis.

	Model	GTD	Theory
Corresponding time	not similar	10 steps = 3 months	-
Mean size of attacks (casualties)	-	similar	similar
Variance size of attacks (casualties)	-	not similar	-
Distribution size of attacks (casualties)	-	not similar	shows Paretian tails
Distribution size of attacks (damage)	not similar	-	not similar
Waiting time	not similar	-	-
Number of attacks	-	-	not similar

Chapter 4

Model extensions

The model Bulleit and Drewek presented has many possibilities for the initial conditions to be taken. We can for example take a different number of initial civilian agents, a different number of resource piles or a different number of tags each non-security agent has. When changing each of the initial conditions each change can be regarded as a different situation to which the model fits. For example using more different types of resource piles might fit a different society better. With each of the changes in the initial conditions there is a possibility that the results will change, and therefore there is the possibility that the model will be better suitable for other situations. Furthermore the model is just one of many models that could have been implemented. In this chapter some adaptations to the model will be implemented and analysed. First a new implementation of the sizes of attacks is implemented and analysed, and after that a new way of movement of non-security ways is described and analysed.

4.1 Pareto distributed planned sizes of attacks

As stated in Chapter 3, the sizes of attacks are one of the weaknesses of the model when comparing it to reality. In order to solve this weakness we will use the results of Clauset and Woodard to give a new implementation of the size of attacks. Clauset and Woodard [13] concluded that the sizes of attacks are power law distributed. The difference in code, which is described in the next section, can be found in Appendix C (page 73).

4.1.1 Description of adapted model

In the original model, described in chapter 2, all attacks had a size of 3×3 nodes. With this implementation we concluded in chapter 3 that the results did mostly not correspond to the GTD. We found that there are too few big attacks in the simulations in comparison with reality. To make the model more comparable to reality we will therefore have to include a way which makes it possible that attacks become bigger. We will make use of the findings of Clauset and Woodard [13], who state that the sizes of terrorist attacks are Pareto distributed with a shape factor of 2.91. We will use these findings in order to generate Pareto distributed random numbers that will be used in order to determine the sizes of the attack in terms of nodes in the environment. Intuitively we know that it is most common to have small sized attacks, this size of attack is the size which was implemented in the original model (3×3 nodes). By using a Pareto random number generator, it is now also possible that an attack covers more nodes, such as 5×5 nodes, 7×7 nodes or in extremely rare cases even more nodes. With this new implementation we expect to see attacks that have more casualties.

As is said in the previous paragraph, the sizes of attacks planned by terrorist agents are now implemented with a random number generator based on the Pareto distribution, and using the shape factor of 2.91 that Clauset and Woodard found. The random number generator is given in equation (4.1) [36]. In this equation U is uniformly distributed between 0 and 1.

$$\frac{1}{U^{\frac{1}{2.91}}} \tag{4.1}$$

The random number calculated by equation (4.1) will be rounded of to the nearest integer. This integer is the radius of the extended Moore neighbourhood which is the neighbourhood in which the attack takes place. An example of the extended Moore neighbourhood is graphically shown in figure 4.1. In this figure the representation of a neighbourhood on the right is the extended Moore neighbourhood, it is represented for different radii: a radius of 1 includes only the darkest colour blue, a radius of 2 is one node larger at each site it includes the darkest and the second darkest colour blue and a radius of 3 is again a node larger at each site it includes also the lightest colour blue drawn. For radii larger than 3 the pattern is continued (every time the radius becomes 1 bigger a node is added at each site) [42].

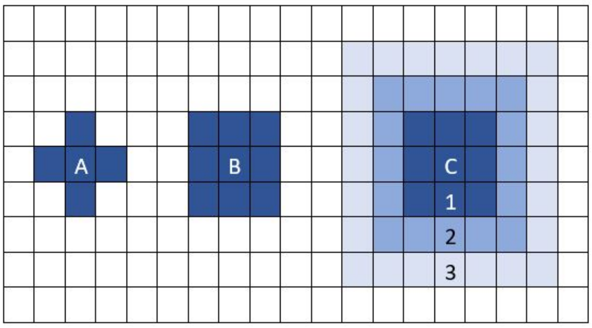

Figure 4.1: Representation of the different neighbourhoods used in the model. Left (Agent A): von Neumann neighbourhood, middle (Agent B): Moore neighbourhood, right (Agent C): extended Moore neighbourhood. Note that in the right representation the neighbourhood is presented for different radii, a radius of 1 includes only the darkest colour blue, a radius of 2 includes the darkest and the second darkest colour blue and a radius of 3 includes also the lightest colour blue drawn. For radii larger than 3 the pattern is continued.

4.1.2 Expectations

In the original model the sizes of attacks in terms of nodes were fixed by 3×3 nodes. In this extended model the sizes of attacks are now at least the same size in terms of nodes of the original model, meaning that sizes of attacks now cover at least 3×3 nodes. It is very likely that there are bigger sized attacks (in terms of the amount of nodes the attack covers). We would therefore expect to see some attacks having more casualties, and therefore also more fear should be created. Furthermore, due to this creation of fear more security agents are expected to be present.

4.1.3 Statistical analysis

For the simulations all other conditions were kept the same. Meaning that besides the explanations in chapter 2 also the following initial conditions as explained in chapter 3 have been used:

- The area is a 50×50 grid,
- Initial amount of civilian agents equal 500,
- Total number of timesteps equal 1030,
- 5 non-security agent tag variables,
- 9 resource piles, with 4 different types of resources. The highest points of the sugarpiles in accordance with Figure 2.1.

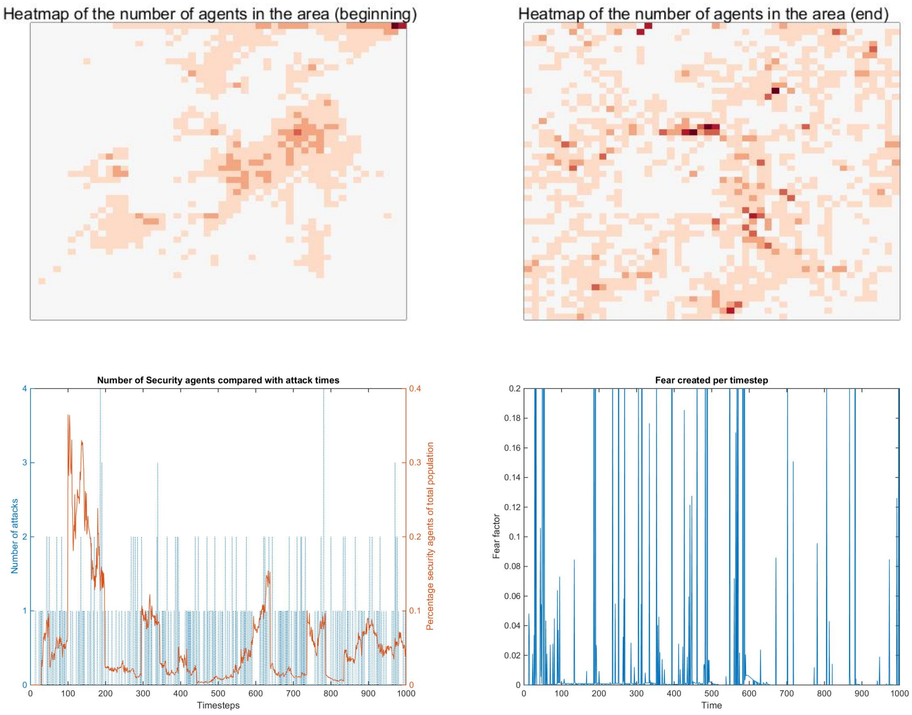

Figure 4.2: Comparable figures of the general analysis in Chapter 3. Top left: Distribution of the number of agents in the simulated community of one of the simulations (first 150 timesteps). Top right: Distribution of the number of agents in the simulated community of one of the simulations (timesteps 850 till 1000). For both top figures: The dark red colour represents relatively many agents passed that node, a white colour represents almost no agents passed the node. Bottom left: Percentage of security agents present in the model (red line), the peaks in the graph correspond to the attacks that produced relatively a lot of fear by the non-security agents. The blue bars show the amount of attacks that have taken place on the timestep. Bottom right: Fear factor that is created per timestep.

When comparing the individual figures of Figure 4.2 with the Figures 3.2, 3.3 and 3.4, we can see that the agents are less scattered at the beginning of the simulation (top left in Figure 4.2), whilst at the end they seem as distributed as in the original model. The end might even be regarded as more clustered than in the original model, since we can see some dark red nodes around the tops of the resource piles (when comparing with Figure 2.1). The distribution of agents at the beginning of the simulations of the original model is visually more convincing as a good model when compared to the top left plot in Figure 4.2. However no information is found on what is right or wrong in these scenario's. The percentage of security agents is higher than in the original mode, and we can also note that the amount of fear produced by terrorist attacks is higher than in the original model. These two observations are in accordance with what we expected to see. From these observations we would expect to have many differences caused by the extension.

This extended model has a small difference with respect to the original model if the time in the simulations is again compared to time in the Global Terrorism Database. Recall that we found that 10 timesteps in the model was comparable to 3 months in the Global Terrorism Database. Testing this comparison with a simple z-test for the extension returns a p-value of 0.0012. With this p-value we should reject the null hypothesis that the mean number of attacks in 10 timesteps equals the mean number of attacks in 3 months of the GTD. The alternative hypothesis states that the means are not similar. A significance level of 0.05 has been used.
Comparing 3 months with 11 timesteps returns a p-value of 0.6313. With this p-value we would not reject the previously stated null hypothesis. We should therefore conclude that in terms of time the model is quite similar, but not exactly the same. The difference we found might be caused by analysing only eleven simulations. Since the difference in time is small, doing more simulations (and using them in the analysis) might have led to a conclusion in which the z-test indicated that both models have a similar division of time.

Casualties

For the extended model the mean number of casualties per attack for the eleven simulations is considered to be different when compared to the mean number of casualties per attack according to the GTD. The simulations of the extended model had on average 19.3 casualties (compared with 13.06 according to the GTD). A z-test shows that the null hypothesis which states that the mean number of casualties per attack for the simulations equal the mean number of casualties per attack for the GTD should be rejected in favour of the alternative hypothesis. The alternative hypothesis stated that these means are different. The p-value that the test returned was $8.7686 \cdot 10^{-16}$ and a significance level of 0.05 was used. This does not mean that the extension does not work, since we should expect the average number of casualties per attack to be higher for the extension as it was for the original model. After all, the attack size in terms of nodes has a possibility to increase and therefore more casualties can be made. A possible way to solve this in another extension is to also make some attacks smaller, and therefore causing the average number of casualties per attack to become lower again. This can be done by for example using the von Neumann neighbourhood instead of the Moore neighbourhood. A visual explanation of the differences of these neighbourhoods can be found in Figure 4.1.

The variances are still regarded to be significantly different. Using the same variance test as with the original mode, the test again returned a p-value of 0.

Using the same multiple comparison test with Bonferonni correction as in section 3.2.1 we can again conclude from Figure 4.3 that the mean number of casualties per attack is very similar in all simulations. However, as concluded in the paragraph above, not we we compare the means between simulations and the GTD. The simulations that differed significantly in mean from the GTD are coloured in red in Figure 4.3.

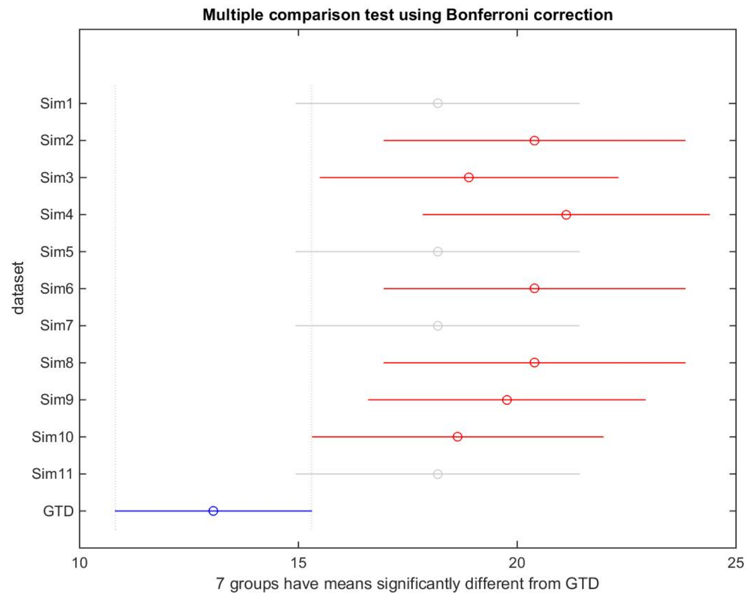

Figure 4.3: Multiple comparison test with all eleven simulations separate and the dataset of casualties from the Global Terrorism Database using the Bonferroni correction. The multiple comparison test tests which datasets are considered to have equal means and which datasets are considered to have different means.

In Figure 4.4 the Quantile-Quantile plot compares the underlying distributions of the combined dataset and the GTD. From this plot we will again have to conclude that the distributions of the combined dataset and of the GTD are not similar.

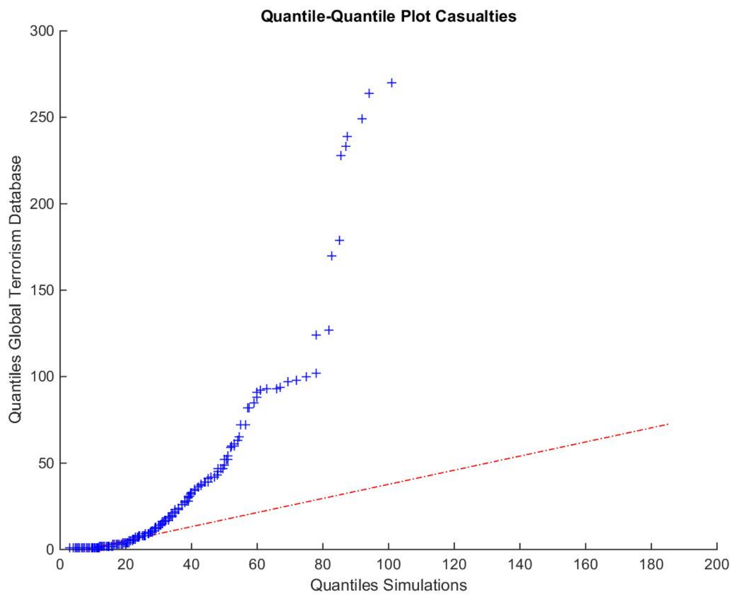

Figure 4.4: Quantile-Quantile plot of the number of casualties per timestep (for the combined dataset of the simulations) and the number of casualties per 3 months for the GTD.

Figure 4.5 shows the histograms of the frequencies of the number of casualties. Also from these histograms we would not expect the datasets to follow the same distribution. Since the number of casualties per attack of the GTD (right histogram) is relatively more often lower than we see in the histogram of the simulations (left histogram).

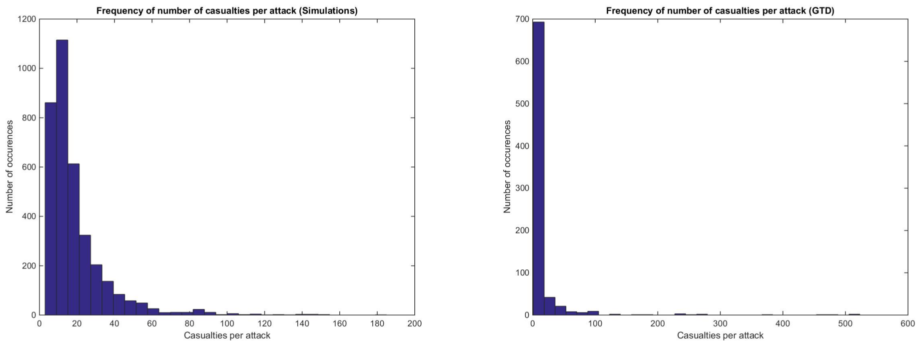

Figure 4.5: Histogram of the frequency of the number of casualties per attack. Left: combined dataset of the simulations. Right: The Global Terrorism Database.

A null hypothesis is used that states that the combined dataset of the simulations is distributed as the GTD and a alternative hypothesis states that these distributions are not the same. The two sample Kolmogorov-Smirnov test rejects the null hypothesis with a p-value of $3.3586 \cdot 10^{-279}$.

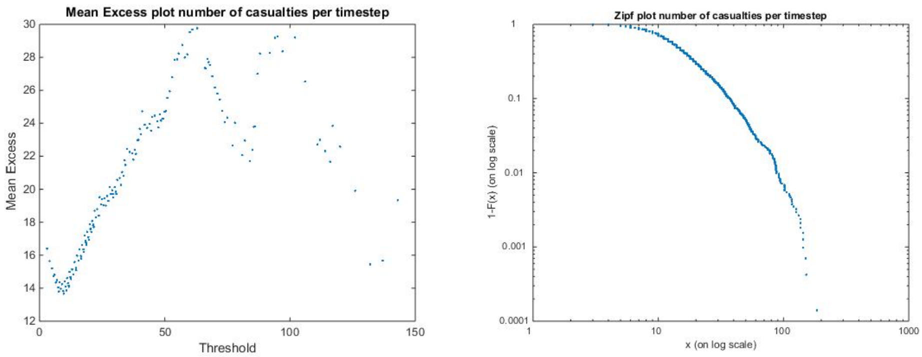

Figure 4.6: Left: Mean Excess plot of the number of casualties per timestep. As can be seen again after a threshold of 10 we can see some starting Paretianity by the linearity of the data. Right: Log-log plot (or Zipf plot) of the empirical survival function. This plot also shows that for more than 10 casualties per timestep the data shows some Paretianity.

The Mean Excess plot and the Zipf plot which are used to analyse whether or not the data follows a Pareto distribution are shown in Figure 3.8. As with the original data, this extension is also likely to have Paretian tails. We derive this from both plots since the Zipf plot (right plot in Figure 3.8) shows linearity of the data for $x > 10$ (where x are the ordered data). Also the clearly upward trend for more than 10 casualties per timestep in the Mean Excess plot (left plot in Figure 3.8) gives us a sign of some Paretianity. Besides the fact that the extension used the Pareto distributed sizes of attacks (in terms of nodes) with a parameter of 2.91 we have to conclude that we are still not satisfied with the tail parameter which is around 6, since we expect a parameter of 2.91, according to Clauset and Woodard [13].

The chi-squared goodness-of-fit test is used, and several parameters have been tried. All parameters returned a p-value of 0. Therefore, the null hypothesis that the combined dataset has a Pareto distribution is rejected in favour of the alternative hypothesis that the combined dataset does not follow such a distribution. A significance level of 0.05 was used.

We will therefore again conclude that the number of casualties is not distributed as the GTD. And the number of casualties show better signs of Paretianity, however it is still not the same parameter of the Pareto distribution as we expect.

Damage

In Figure 4.7 the multiple comparison test with Bonferonni correction is displayed on the left. As we can notice in comparison with Figure 3.9 the simulations are as consistent with each other as in the original model. Again we can see one very outstanding simulation: simulation 9 differs in mean significantly from simulations 2, 6 and 8. The histogram on the right in Figure 4.7 shows the frequency of the destroyed resource values of the combined dataset. As we can see this is similar to Figure 3.10. However, we now also see that some attacks destroyed bigger resource values, this is most likely due to the new implementation of a possibility of bigger attacks.

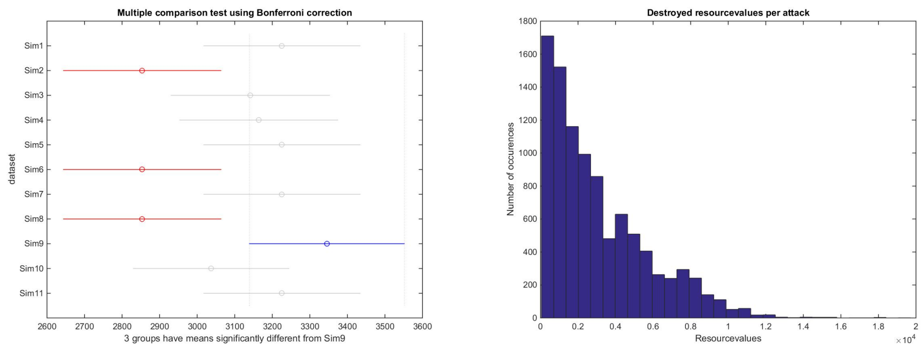

Figure 4.7: Left: Multiple comparison test with Bonferroni correction for the data on destroyed resource values of each of the simulations. As can be noticed e.g. simulations 4, 5, 6, 9 and 10 are considered to have a different mean from simulation 2. Also other simulations are considered to have different means. Right: The histogram of the destroyed resource values of the combined dataset.

For the original model we noticed that the Weibull distribution best corresponds (but does not correspond well enough) with the distribution of the destroyed resource values. Figure 4.8 shows the Quantile-Quantile plot of the destroyed resource values in the combined dataset and the theoretical quantiles of the Weibull distribution. As we can see the Weibull distribution fits the data better than the data of the original model. However, we do still not see a quite perfect straight line, the tail of the data is significantly different than the theoretically expected values.

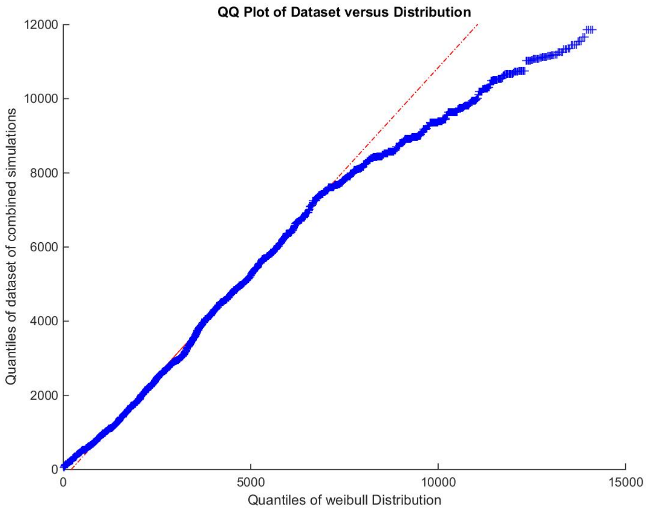

Figure 4.8: Quantile-Quantile plot of the quantiles of resource values and the theoretical quantiles of the Weibull distribution. It shows a better fit of the Weibull distribution than it fitted the data of the original model. However, there is still a significant difference.

The chi-squared goodness-of-fit test tells us with a p-value of $8.6928 \cdot 10^{-40}$ to reject the null hypothesis in favour of the alternative hypothesis. The null hypothesis used is that the combined dataset of resource values follows a Weibull distribution, whereas the alternative hypothesis states that it does not follow such a distribution. The significance level used is 0.05.

Waiting time

In the analysis of the data of the original model we noted that visually the waiting times were exponentially distributed, whilst the conservative tests showed it was not. Figure 4.9 shows on the left the histogram, which gives us the assumption that our data is again exponentially distributed. The red line in the histogram is the exponential fit. The Quantile-Quantile plot on the right in the same figure, shows that the exponential distribution fits less than in the original model, since the straight line deviates quite a bit in the tail.

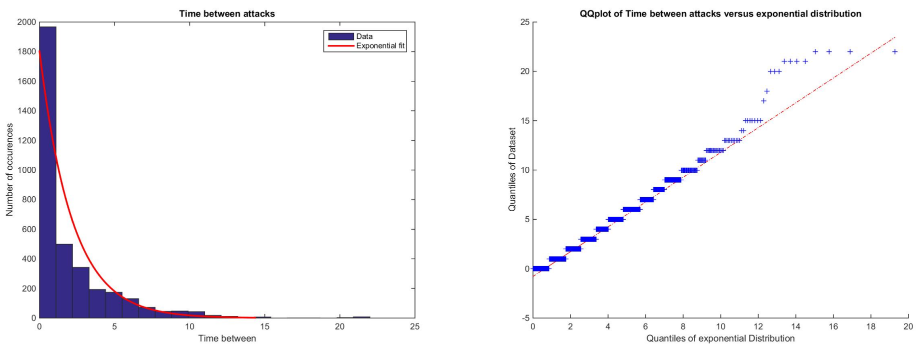

Figure 4.9: Left: Histogram of the waiting time between the attacks for the combined dataset. It suggests an exponential distribution, as the red line indicates (fitted exponential distribution). Right: Quantile-Quantile plot of the combined dataset of destroyed resource values and the theoretical quantiles of the exponential distribution.

Furthermore, the chi-squared goodness-of-fit test rejects the null hypothesis that the waiting times are exponentially distributed in favour of the alternative hypothesis that they are not distributed in such a way (p-value $5.1394 \cdot 10^{-46}$). Also the Kolmogorov-Smirnov test rejects the same null hypothesis (p-value 0). Based on our visual findings, we might have to stop believing that the waiting times of the extended model are exponentially distributed. This might be caused by the implementation that a few attacks become bigger. Bigger attacks destroy more agents, and it might be the case that relatively too many security agents are destroyed which causes less terrorist agents to be arrested and therefore the attacks to have more time between them with respect to the original model. As we can notice in the Quantile-Quantile plot of Figure 4.9 the data points (blue crosses) in the tail of the figure are above the red line. This means that the quantiles of the exponential distribution expects them to be in a lower quantile of the dataset. Meaning that the waiting time in some cases is theoretically too long. Therefore, this might be one of the weaknesses of the extended model, when compared to the original model.

Number of attacks

Figure 4.10 shows the number of attacks per timestep. We would expect the number to be evenly distributed over time.

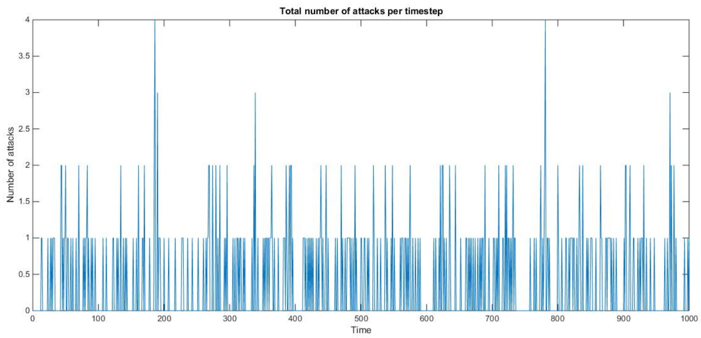

Figure 4.10: Number of attacks over 1000 timesteps for one of the simulations.

The chi-squared goodness-of-fit test does not reject the null hypothesis that the number of attacks are a homogeneous Poisson process. Therefore the number of attacks is on average constant over time if we split time into non-overlapping bins of 80 timesteps. The alternative hypothesis stated that the number of attacks do not follow a homogeneous Poisson process. The p-value was 0.7071.

Furthermore the Multinomial test with 'Monte Carlo approach' does reject the null hypothesis with a p-value of 0. The null hypothesis and alternative hypothesis were the same as with the chi-squared goodness-of-fit test stated above.

From our visual findings and the chi-squared goodness-of-fit test we would conclude that the number of attacks follow a homogeneous Poisson process.

4.2 Different implementation of moving agents

Another change in the implementation of the model of Bulleit and Drewek that has been tried is to change the way non-security agents move. This extension is independently implemented of the extension described in the previous section. The reason for trying a change of movement is that we have seen in the analysis in Chapter 3 that the distributions of the resources and agents destroyed by the attacks are not what theory, the Global Terrorism Database and/or the model of Bulleit and Drewek expected. By trying a different and more realistic way the agents move across the area we hope that this solves at least some of the problems. The difference in code, which is explained in the following sections, can be found in Appendix D (page 75).

4.2.1 Description of adapted model

In the original implemented model, described in Chapter 2 all agents moved to a node in their Moore neighbourhood. Since the model has discrete timesteps, it is realistic to think that agents could also move a few nodes at the time. In real life there are always people that move faster, or some types of transport goes faster than another. In order to implement this difference the vision of the civilian and (un)active terrorist agents has been used. In this extended model every non-security agent can move up to the amount of nodes in its vision. Recall that every non-security agents vision is determined at the start of the model or at the time the agent is created. The vision is uniformly distributed between 1 and 5. This new implementation causes every non-security agent to move in their extended Moore neighbourhood with a radius equal to the agent's vision. An example of the extended Moore neighbourhood is graphically shown in Figure 4.1.

4.2.2 Expectations

Taking this extended version of the model into account with respect to the original implemented model, we would expect the non-security agents to have an higher Generalized Resource value, which would mean that the average Generalised Wealth would be higher in this adapted model. Since the Generalised Wealth is higher the destroyed resource value per attack is expected to show some higher values. Furthermore, since non-security agents are able to choose more nodes where they want to go, and since this causes the Generalised Wealth to be higher we would expect the number of (un)active terrorist agents to be higher. This we would expect since with a larger Generalised Wealth it is easier to satisfy the conditions to become an (un)active terrorist agent.

4.2.3 Statistical analysis

For the simulations all other conditions were kept the same. Meaning that besides the explanations in Chapter 2 also the following initial conditions as explained in Chapter 3 have been used:

- The area is a 50×50 grid,
- Initial amount of civilian agents equal 500,
- Total number of timesteps equal 1030,
- 5 non-security agent tag variables,
- 9 resource piles, with 4 different types of resources. The highest points of the sugarpiles in accordance with Figure 2.1.

Figure 4.11: Comparable figures of the general analysis in Chapter 3. Top left: Distribution of the number of agents in the simulated community of one of the simulations (first 150 timesteps). Top right: Distribution of the number of agents in the simulated community of one of the simulations (timesteps 850 till 1000). For both top figures: The dark red colour represents relatively many agents passed that node, a white colour represents almost no agents passed the node. Bottom left: Percentage of security agents present in the model (red line), the peaks in the graph correspond to the attacks that produced relatively a lot of fear by the non-security agents. The blue bars show the amount of attacks that have taken place on the timestep. Bottom right: Fear factor that is created per timestep.

When comparing the individual figures of Figure 4.11 with the Figures 3.2, 3.3 and 3.4, we can see that the agents are less distributed in the beginning of the simulation (top left in Figure 4.11), whilst at the end they seem as distributed as in the original model. The distribution of agents in the beginning of the simulations of the original model is visually more convincing as a good model when compared to the top left plot in Figure 4.11. However no information is found on what is right or wrong in these scenario's. Also the number of security agents and the fear factor seem to be more or less the same as in the original model. From these observations we would not expect to have many differences caused by the extension.

This extended model has a small difference with respect to the original model if the time in the simulations is again compared to time in the Global Terrorism Database. Recall that in the original model we found that 10 timesteps in the model was comparable to 3 months in the Global Terrorism Database. Testing this comparison with a simple z-test for the extensions returns a p-value of 0.0352. And with this p-value we should reject the null hypothesis that the mean number of attacks in 10 timesteps equals the mean number of attacks in 3 months of the GTD. The alternative hypothesis states that the means are not similar. A significance level of 0.05 has been used.
Comparing 3 months with 9 timesteps returns a p-value of 0.2245. With this p-value we would

not reject the previously stated null hypothesis. We should therefore conclude that in terms of time the model is similar, but not exactly the same. The difference we found might be caused by analysing only eleven simulations. Since the difference in time is very small, doing more simulations (and using them in the analysis) might have led to a conclusion in which the z-test indicated that both models have a similar division of time.

Casualties

The mean number of casualties per attack for the eleven simulations is still considered to not be different when compared to the mean number of casualties per attack according to the GTD. The simulations of the extended model had on average 14.3 casualties (compared with 13.06 according to the GTD). A z-test shows that the null hypothesis which states that the mean number of casualties per attack for the simulations equal the mean number of casualties per attack for the GTD does not have to be rejected. The alternative hypothesis stated that these means are different. The p-value that the test returned was 0.0729. And a significance level of 0.05 was used.

The variances are still regarded to be significantly different. Using the same test as with the original mode, the test again returned a p-value of 0.

With the same multiple comparison test with Bonferonni correction as in section 3.2.1 we can again conclude from Figure 4.12 that the mean number of casualties per attack is very similar in all simulations and also when compared with the GTD.

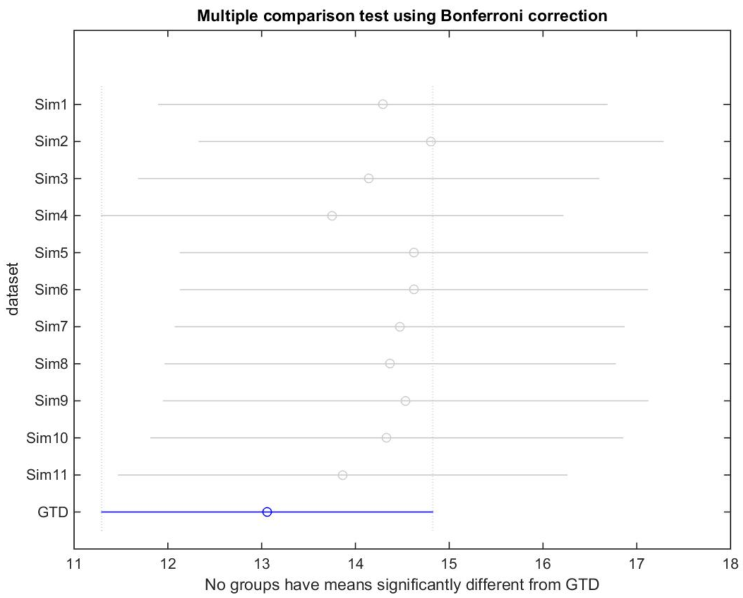

Figure 4.12: Multiple comparison test with all eleven simulations separate and the dataset of casualties from the Global Terrorism Database using the Bonferroni correction. The multiple comparison test tests which datasets are considered to have equal means and which datasets are considered to have different means.

In Figure 4.13 the Quantile-Quantile plot compares the underlying distributions of the combined dataset and the GTD. From this plot we will again have to conclude that the distributions are not similar.

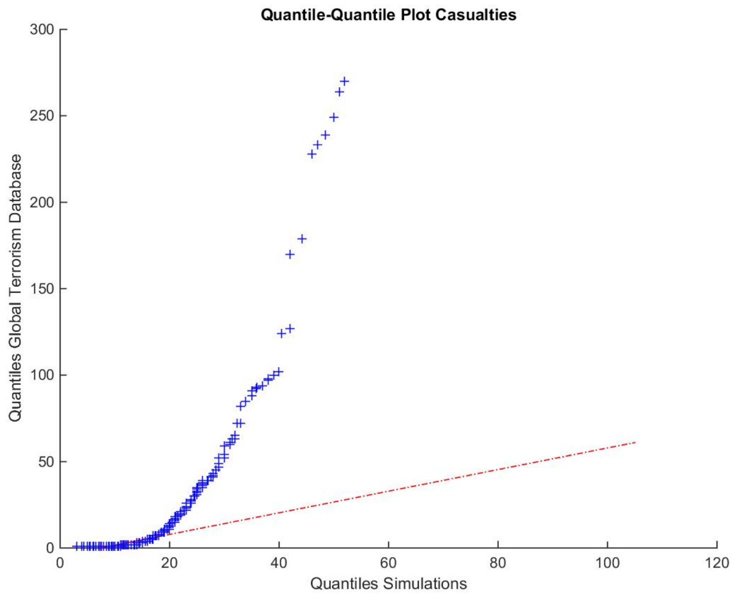

Figure 4.13: Quantile-Quantile plot of the number of casualties per timestep (for the combined dataset of the simulations) and the number of casualties per 3 months for the GTD.

in Figure 4.14 the histograms of the frequencies of the number of casualties have been displayed. Also from these histograms we would not expect the datasets to follow the same distribution, since the number of casualties per attack of the GTD (right histogram) is more often lower than we see in the histogram of the simulations (left histogram).

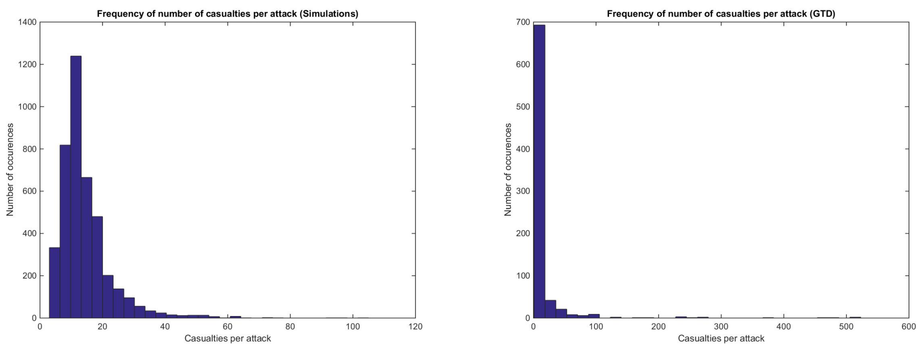

Figure 4.14: Histogram of the frequency of the number of casualties per attack. Left: combined dataset of the simulations. Right: The Global Terrorism Database.

With a null hypothesis that states that the combined dataset of the simulations is distributed as the GTD and as alternative hypothesis that these distributions are not the same, the Kolmogorov-Smirnov test rejects the null hypothesis with a p-value of $6.0071 \cdot 10^{-294}$.

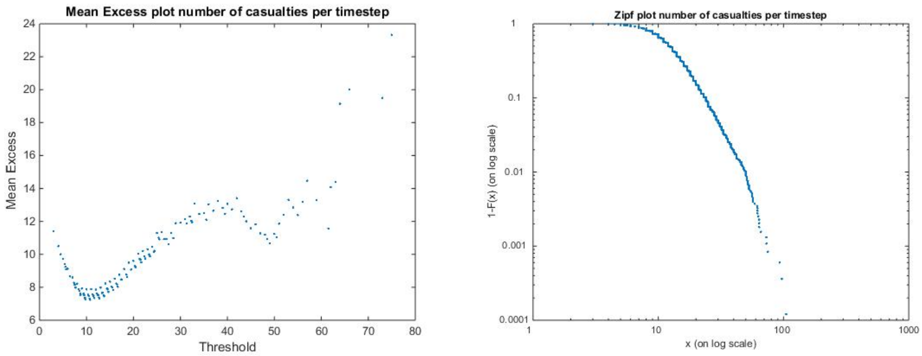

Figure 4.15: Left: Mean Excess plot of the number of casualties per timestep. As can be seen again after a threshold of 15 we can see some starting Paretianity by the linearity of the data. Right: Log-log plot (or Zipf plot) of the empirical survival function. This plot also shows that for more than 15 casualties per timestep the data shows some Paretianity.

The Mean Excess plot and the Zipf plot which are used to analyse whether or not the data follows a Pareto distribution are shown in Figure 4.15. Compared to the original model, this extension also shows some Paretianity in the tails. We derive this from both plots since the Zipf plot (right plot in Figure 4.15) shows linearity of the data for $x > 15$ (where x are the ordered data). Also the clearly upward trend for more than 15 casualties per timestep in the Mean Excess plot (left plot in Figure 4.15) gives us a sign of some Paretianity. We still have to conclude that we are not satisfied with the tail parameter which is around 8, since we expect a parameter of 2.91, according to Clauset and Woodard [13].

The chi-squared goodness-of-fit test is used. And with a p-value of 0 the null hypothesis that the combined dataset has a Pareto distribution is rejected in favour of the alternative hypothesis that the combined dataset does not follow such a distribution. In both tests a significance level of 0.05 was used. Several parameters for the Pareto distribution have been tried, but none resulted in a better p-value.

As said with the analysis of the original model with statistical evidence it is hard to prove whether or not empirical data follows a Pareto Distribution due to large fluctuations in the tail. From the Zipf plot and the Mean Excess plot in Figure 4.15 we can clearly see that the tail of the data shows some Paretianity. But it still is one of the weaknesses of the model.

Damage

In Figure 4.16 the multiple Comparison test with Bonferonni correction is displayed on the left. As we can notice in comparison with Figure 3.9 the simulations are less consistent with each other. This means that the average destroyed resource value per attack differs significantly between multiple simulations, and differs even more than we already noticed for the original model. Due to this difference of outcomes between the various simulations, the extended model might in this part be worse than the original model. The histogram on the right in Figure 4.16 shows the frequency of the destroyed resource values of the combined dataset. As we can see this is similar to Figure 3.10.

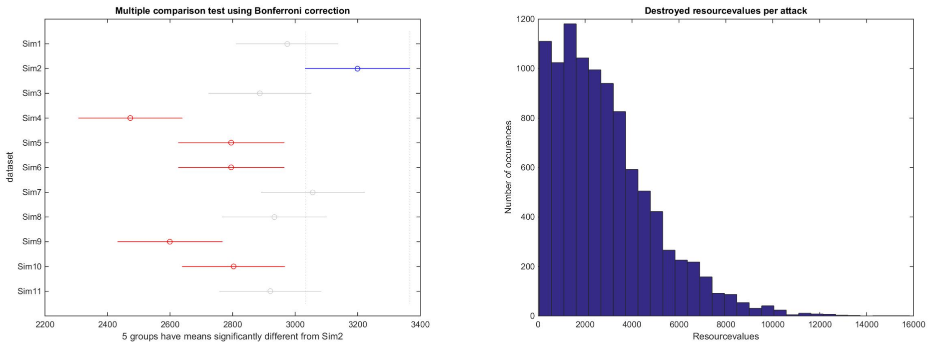

Figure 4.16: Left: Multiple comparison test with Bonferroni correction for the data on destroyed resource values of each of the simulations. As can be noticed e.g. simulations 4, 5, 6, 9 and 10 are considered to have a different mean from simulation 2. Also other simulations are considered to have different means. Right: The histogram of the destroyed resource values of the combined dataset.

For the original model we noticed that the Weibull distribution best corresponds (but does not correspond well enough) with the distribution of the destroyed resource values. Figure 4.17 shows the Quantile-Quantile plot of the destroyed resource values in the combined dataset and the theoretical quantiles of the Weibull distribution. As we can see the data is in a quite good straight line. We would therefore expect the data to be Weibull distributed. This is a good improvement compared to the original model. Furthermore, we expected the average value of resources destroyed to be higher in this extended model. As we can see in the histogram on the right in Figure 4.16, the destroyed resource values is more or less the same as in the original model.

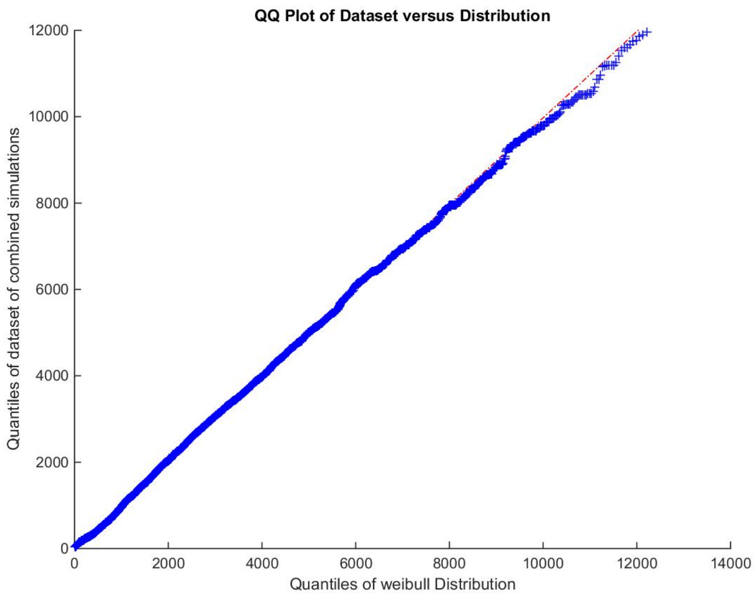

Figure 4.17: Quantile-Quantile plot of the quantiles of resource values and the theoretical quantiles of the Weibull distribution. It shows a near perfect straight line. Meaning that the destroyed resource values are likely to be Weibull distributed.

Also the chi-squared goodness-of-fit test tells us with a p-value of 0.0905 not to reject the null hypothesis. The null hypothesis used is that the combined dataset of resource values follows a Weibull distribution, whereas the alternative hypothesis states that it does not follow such a distribution. The significance level used is 0.05. It should be noted that with the same null and alternative hypothesis the Kolmogorov-Smirnov test (p-value $5.3398 \cdot 10^{-04}$) and the Anderson-Darling test (p-value $5.0000 \cdot 10^{-04}$) reject the null hypothesis, but these tests are more conservative and based on continuous distributions. Therefore the chi-squared goodness-of-fit test is a better test for this data.

Waiting time

In the analysis of the data of the original model we noted that visually the waiting times were exponentially distributed, whilst the conservative tests showed it was not. Figure 4.18 shows on the left the histogram, this gives us the expectation that our data is moving towards a generalisation of the exponential distribution, which is the Weibull distribution. The red line in the histogram is the exponential fit. The Quantile-Quantile plot on the right in the same figure, shows that the exponential distribution fits less than in the original model, since the straight line deviates quite a bit.

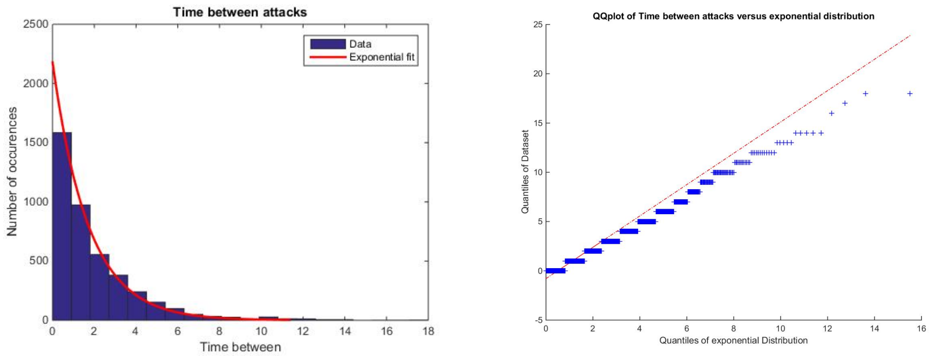

Figure 4.18: Left: Histogram of the waiting time between the attacks for the combined dataset. It suggests an exponential distribution, as the red line indicates (fitted exponential distribution). Right: Quantile-Quantile plot of the combined dataset of destroyed resource values and the theoretical quantiles of the exponential distribution.

Furthermore, the chi-squared goodness-of-fit test rejects the null hypothesis that the waiting times are exponentially distributed in favour of the alternative hypothesis that they are not distributed in such a way (p-value $2.1503 \cdot 10{-}35$). Also the Kolmogorov-Smirnov test rejects the same null hypothesis (p-value 0). Based on our visual findings, we believe that our data is moving towards a generalisation of the exponential distribution. The original model showed a better relationship between the waiting times and the exponential distribution.

Number of attacks

Figure 4.19 shows the number of attacks per timestep. We would expect the number to be evenly distributed over time.

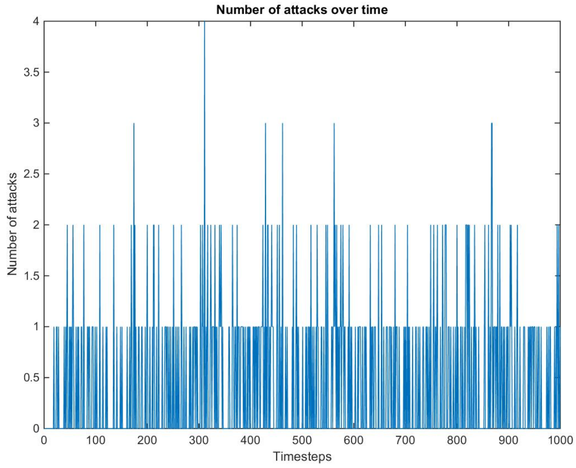

Figure 4.19: Number of attacks over 1000 timesteps for one of the simulations.

The chi-squared goodness-of-fit test does not reject the null hypothesis that the number of attacks are a homogeneous Poisson process. Therefore the number of attacks is on average constant over time if we split time into non-overlapping bins of 40 timesteps. The alternative hypothesis stated that the Number of attacks do not follow a homogeneous Poisson process. The p-value was 0.4930. We should note that the chi-squared goodness-of-fit test (p-value 0.0187) rejected this null hypothesis when 80 timesteps per bin were used.

Furthermore the Multinomial test with 'Monte Carlo approach' rejects the null hypothesis with a p-value of 0. The null hypothesis and alternative hypothesis were the same as with the chi-squared goodness-of-fit test stated above.

From our visual findings and from the chi-squared goodness-of-fit test we can conclude that the number of attacks follow a homogeneous Poisson process.

4.3 Conclusion

The first extension, the implementation of different planned sizes of attacks, did not have many improvements. In the analysis the expected rise in percentage security agents and rise in fear values has been noticed. However the extended model could not conclude that the waiting times are exponentially distributed, whereas the original model could. On the other hand the Weibull distribution better fits the data on destroyed resource values than the data of the original model, but still differs significantly in the tail. Due to the lack of big improvements in the various distributions we should conclude that this extension is not favourable for the model.

The second extension, which implemented that non-security agents could skip nodes when moving, had some improvement when analysed. The destroyed resource values are now believed to follow a Weibull distribution and the number of attacks are believed to be a homogeneous Poisson process. On the other hand the waiting times seemed to be less exponentially distributed and the mean destroyed resource values per simulation differed more than in the original model. However since we have an improvement in the fit of the Weibull distribution to the data of the destroyed resource values and since the number of attacks are now, as expected, a homogeneous Poisson process we might be able to use this extension in future versions of the model.

Chapter 5

Conclusion and recommendations

With the model discussed we got insight in the phenomena relating to terrorist attacks. We found that the model and its extensions discussed are not completely comparable to reality. For the original model we found that the mean number of casualties per attack of the simulations equal the mean of the Global Terrorism Database and that the waiting times are exponentially distributed. The extension which implemented a different way in which the non-security agents moved also shows that the destroyed resource values are Weibull distributed and that the number of attacks are a Poisson process.

Shortcomings of the (extended) model are still that the number of casualties only show signs of Paretianity. Therefore, from the analysis of the original model and its discussed extensions we can conclude that further research and modifications are necessary to be able to state that the model is comparable to reality. Whilst the original model is quite consistent in the results it gives and since the waiting times are exponentially distributed, it needs further extensions in order for the model to become comparable to reality. These extensions should especially focus on the casualties that are caused by the attacks.
It is suggested that the model is improved by implementing more realistic features, such as a dependency of the number of terrorist agents on the security agents, trying different set-ups of the community resource piles or implementing different kinds of attacks. These features have not been analysed, but might make the data of the model even more comparable to reality.

Furthermore, we should remark that only eleven simulations have been used in the analysis of the three models discussed (original model, extended model 1 and extended model 2). Eleven simulations is not significantly many for statistical analysis. Furthermore, it should be kept in mind that using more simulations could make the conclusions as drawn in this report more clear, or it could change them. It is therefore strongly advised to use more simulations for further analysis and for better conclusions.

Although we are not 100% satisfied with the outcomes, we can use the model to analyse security policies. We can for example analyse the consequences of a new type of security. This is especially valuable in the understanding of terrorist activity in relationship with different security policies.

Bibliography

[1] Axelrod, R. (1997). The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration. Princeton, NJ: Princeton University Press.

[2] Axelrod, R., & Tesfatsion, L. (2017, April 25). On-Line Guide for Newcomers to Agent-Based Modeling in the Social Sciences. Retrieved June 05, 2017, from http://www2.econ.iastate.edu/tesfatsi/abmread.htm

[3] Bloomberg News. (2017, May 23). Recent Major Terror Attacks in Europe. Retrieved May 30, 2017, from https://www.bloomberg.com/news/articles/2017-05-23/recent-major-attacks-in-europe

[4] Bulleit, W. M., & Drewek, M. W. (2012). Agent-Based Modeling and Simulation for Hazard Management. Retrieved from https://www.researchgate.net/publication/266461234_Agent-Based_Modeling_and_Simulation_for_Hazard_Management

[5] Bulleit, W. M., & Drewek, M. W. (2011). Agent-Based Simulation for Human-Induced Hazard Analysis. Risk Analysis, 31(2), 205-217. doi:10.1111/j.1539-6924.2010.01497.x

[6] Bulleit, W. M., & Drewek, M. W. (2005). Simulating Initial Conditions In Agent-Based Modeling (Proceedings of the Agent2005 Conference). Retrieved by e-mail

[7] Bulleit, W. M., & Drewek, M. W. (n.d.). An Agent-Based Model of Terrorist Activity. Retrieved from https://pdfs.semanticscholar.org/22eb/a1ad37399585c9e007cfcdf7dca7afe6f063.pdf

[8] Bulleit, W. M., & Drewek, M. W. (n.d.). Simulating Terrorism in a Community. Retrieved by e-mail

[9] Cederman, L. E. (2003). Generating State-Size Distributions: A Geopolitical Model. Retrieved from https://icr.ethz.ch/publications/cederman2003generating.pdf

[10] Cederman, L. E. (2002). Modeling the Size of Wars From Billiard Balls to Sandpiles. Retrieved from http://econ2.econ.iastate.edu/tesfatsi/LarsErikCederman.ModelingSizeOfWars.pdf

[11] Chen, H., Chung, W., Xu, J. J., Wang, G., Qin, Y., & Chau, M. (2004). Crime Data Mining: A General Framework and Some Examples. Computer, 34(4), 50-56. Retrieved from http://ieeexplore.ieee.org/document/1297301/?part=1

[12] Cirillo, P. (2013). Are your data really Pareto distributed? Physica A: Statistical Mechanics and its Applications, 392(23), 5947-5962. Retrieved from https://doi.org/10.1016/j.physa.2013.07.061

[13] Clauset, A., & Woodard, R. (2013). Estimating the Historical and Future Probabilities of Large Terrorist Events. Annals of Applied Statistics, 7(4), 1838-1865. doi:10.1214/12-AOAS614

[14] De Graaf, B. (speaker). (2017, May 25). Spionage [TV Show]. DWDD University. Hilversum, The Netherlands: VARA. Retrieved from https://www.npo.nl/dwdd-university-presenteert-spionage/25-05-2017/VARA_101383505

[15] Embrechts, P., Klüppelberg, C., & Mikosch, T. (2012). Modelling Extremal Events: for Insurance and Finance (9th ed.). Heidelberg, Germany: Springer.

[16] Epstein, J. M. (2002). Modeling civil violence: An agent-based computational approach. PNA, 99(3), 7243-7250. Retrieved from https://www.cs.helsinki.fi/u/ahyvarin/teaching/niseminar3/papers/Epstein02SimulationOfCivilViolence.pdf

[17] Epstein, J. M., & Axtell, R. (1996). *Growing Artificial Societies: Social Science from the Bottom up*. Washington, DC: Brookings Institution.

[18] Eurostat. (2017, May 17). Statistics on European cities. Retrieved May 24, 2017, from http://ec.europa.eu/eurostat/statistics-explained/index.php/Statistics_on_European_cities

[19] Frykberg, E. R., & Tepas, J. J. (1988). Terrorist Bombings: Lessons Learned From Belfast to Beirut. Ann. Surg., Nov., 569-576. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1493790/pdf/annsurg00189-0043.pdf

[20] Gompertz, B. (1825). On the Nature of the Function Expressive of the Law of Human Mortality, and on a New Mode of Determining the Value of Life Contingencies. Philosophical Transactions of the Royal Society of London, Vol. 115, 513-583. Retrieved from http://www.jstor.org/stable/107756

[21] Grimmett, G., & Welsh, D. (2014). Probability an Introduction (2nd ed.). Oxford, United Kingdom: Oxford University Press.

[22] Gustafsson, L., & Sternad, M. (2010). Consistent micro, macro and state-based population modelling. Mathematical Biosciences, 225(2), 94-107. Retrieved from http://www.sciencedirect.com/science/article/pii/S0025556410000283

[23] Rosenthal, J. S. (2006). A First Look At Rigorous Probability Theory (2nd ed.). Singapore: World Scientific Publishing Co. Pte. Ltd.

[24] Lustick, I. S. (2000, January 31). Agent-based modelling of collective identity: testing constructivist theory. Retrieved from http://jasss.soc.surrey.ac.uk/3/1/1.html

[25] Macal, C. M., & North, M. J. (2006, November 29). Introduction to Agent-based Modeling and Simulation [College-slides]. Retrieved from http://www.mcs.anl.gov/~leyffer/listn/slides-06/MacalNorth.pdf

[26] Macy, M. W., & Willer, R. (2001). From Factors to Actors: Computational Sociology and Agent-Based Modeling. Retrieved from http://www2.econ.iastate.edu/tesfatsi/Macy_Factors_2001.pdf

[27] Mathworks. (z.j.). Generalized Pareto Distribution. Retrieved from http://nl.mathworks.com/help/stats/generalized-pareto-distribution.html

[28] National Consortium for the Study of Terrorism and Responses to Terrorism (START). (2016). Global Terrorism Database (Codebook). Retrieved from https://www.start.umd.edu/gtd/downloads/Codebook.pdf

[29] National Consortium for the Study of Terrorism and Responses to Terrorism (START). (2016). Global Terrorism Database (Entire GTD Dataset). Retrieved from https://www.start.umd.edu/gtd

[30] Rice, J. A. (2007). Mathematical Statistics and Data Analysis (3rd ed.). Belmont, CA: Brooks/Cole.

[31] Sandler, T. (2014). The analytical study of terrorism: Taking stock. Journal of Peace Research, 51(2), 257-271. doi:10.1177/0022343313491277

[32] Santifort-Jordan, C., & Sandler, T. (2014). An Empirical Study of Suicide Terrorism:A Global Analysis. Southern Economic Association, 80(4), 981-1001. doi:10.4284/0038-4038-2013.114

[33] Scharpf, A., Schneider, G., Nöh, A., & Clauset, C. (2014). Forecasting the Risk of Extreme Massacres in Syria. Retrieved from https://pdfs.semanticscholar.org/c9b4/a046753e998713c3259a16429411fffcb1cf.pdf

[34] Schelling, T. (1978). Micromotives and Macrobehavior. New York, NY: W.W. Norton.

[35] Schmid, A. P. (2011). The Routledge Handbook of Terrorism Research. New York, NY: Routledge.

[36] Tanizaki, H. (2004). Computational methods in statistics and econometrics. New York, NY: Marcel Dekker, Inc.

[37] The MathWorks, Inc. (2011). MatLab Student Version with Simulink, Version r2014b [Software]. Retrieved from http://www.mathworks.com/

[38] Tutun, S., Khasawneh, M. T., & Zhuang, J. (2017). New framework that uses patterns and relations to understand terrorist behaviors. Expert Systems with Applications, 78, 358-375. Retrieved from http://www.sciencedirect.com/science/article/pii/S0957417417301161

[39] United Nations, Department of Economic and Social Affairs, Population Division. (2015). World Population Prospects: The 2015 Revision. Retrieved from https://esa.un.org/unpd/wpp/Publications/Files/Key_Findings_WPP_2015.pdf

[40] Virkar, Y., & Clauset, A. (2014). Power-law distributions in binned empirical data. Annals of Applied Statistics, 8(1), 89-119. doi:10.1214/13-AOAS710

[41] Walter, C. (2003). Defining Terrorism in National and International Law. Retrieved from https://www.unodc.org/tldb/bibliography/Biblio_Terr_Def_Walter_2003.pdf

[42] Weisstein, E. W. (n.d.). Moore Neighborhood. Retrieved from http://mathworld.wolfram.com/MooreNeighborhood.html

[43] Weisstein, E. W. (n.d.). von Neumann Neighborhood. Retrieved from http://mathworld.wolfram.com/vonNeumannNeighborhood.html

[44] Wikipedia. (2017, June 7). Gompertz function. Retrieved June 14, 2017, from https://en.wikipedia.org/wiki/Gompertz_function

[45] Wikipedia. (2017, June 13). Weibull distribution. Retrieved June 16, 2017, from https://en.wikipedia.org/wiki/Weibull_distribution

[46] Wikipedia. (2017, May 24). Log-normal distribution. Retrieved June 16, 2017, from https://en.wikipedia.org/wiki/Log-normal_distribution

Appendices

A - Explanation of distributions

Generalized Pareto Distribution

The Generalized Pareto Distribution is a general form of the Pareto distribution wich was introduced by Vilfredo Pareto. A random variable X is said to be follow a Generalized Pareto distribution if it has a cumulative distribution function similar to

$$F_\xi(z) = \begin{cases} (\xi z + 1)^{-\frac{\xi+1}{\xi}} & \text{for } \xi \neq 0 \\ e^{-z} & \text{for } \xi = 0 \end{cases}$$

In the cumulative distribution function $z = \frac{x-\mu}{\sigma}$ and ξ is the shape parameter, μ the location parameter and σ the scale parameter.
Note that if $\xi = \mu = 0$ then the Generalized Pareto Distribution is equal to the exponential Distribution. And if $\xi > 0$ and $\mu = \frac{\sigma}{\xi}$ than the Generalized Pareto Distribution is equal to the Pareto Distribution with scale $x_m = \frac{\sigma}{\xi}$ and shape $\alpha = \frac{1}{\xi}$. [12][27]

Exponential Distribution

A random variable X is said to follow an exponential distribution if it has a cumulative distribution function similar to

$$F_\lambda(x) = \begin{cases} 1 - e^{-\lambda x} & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$$

In which λ is the inverse scale factor.
Note that the exponential distribution describes the time between events in a Poisson process. Furthermore, it is specific case of the Gamma distribution. [23][30]

Weibull Distribution

A random variable X is said to follow a Weibull distribution if it has a cumulative distribution function similar to

$$F_{\lambda,k}(x) = \begin{cases} 1 - e^{-(\frac{x}{\lambda})^k} & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$$

In which $k > 0$ is the shape parameter and $\lambda > 0$ is the scale parameter.
Note that for $k = 1$ we have the exponential distribution. [15][45]

Power law distribution

Power law distributions are distributions for which

$$f(x) \propto L(x)x^{-\alpha}$$

For which it holds that $\lim\limits_{x \to \infty} \frac{L(cx)}{L(x)} = 1$ with $c > 0$ constant. [12][15]

Log-normal Distribution

A random variable X is said to follow a Weibull distribution if it has a cumulative distribution function similar to

$$F_{\mu,\sigma}(x) = \frac{1}{2} + \frac{1}{2} erf\left(\frac{\log(x) - \mu}{\sqrt{2}\sigma}\right)$$

In which $erf(x)$ is the Gauss error function and μ the mean and σ the standard deviation of X.
note that if X is a log-normally distributed random variable (with mean μ and standard deviation σ) then the logarithm of X is normally distributed. [21][46]

B - Code original model

On the next pages a print of the used code of the original model described in Chapter 2 can be found. Note that the code should be read as pages from left to right and then from top till bottom. All functions have been saved as separate Matlab code files. And the first two 'pages' are the main file of the program.

```matlab
%%% Initial conditions
versionnumber = 7;
TotalTimesteps = 50;
TotalGroupTimesteps = 21;
dim = 50;
Numberofdifferentbuildings = 4;
Numberofbuildings = 9;
amountofcivilians = 500;
CivilianTagVariables = 5;
CivilianMemoryVariables = 3+3*Numberofdifferentbuildings+1;
Pb = 0.03;
Pic = 0.03;
TerroristInfoVariables = 8;
Maxcurve = 1875;
MaxSlope = 1;
%%% Global Variables
global AreaMatrix
global BuildingMatrix
global BuildingMatrix2
global ResourceMatrix
global ResourceInfoMatrix
global CivilianInfoMatrix
global CivilianMemoryMatrix
global FearMatrix
global TerroristInfoMatrix
global attacks
global NumberofSecurityAgents
global arrests
global AnalysisMatrix
global Timecounter
%%% Creation Matrices
AreaMatrix = zeros(dim,dim,amountofcivilians, TotalTimesteps);↙
%dim x dim x maximum civilians on one place Area, info stored per timestep
BuildingMatrix = zeros(dim,dim,Numberofdifferentbuildings, TotalTimesteps);           %↙
Distribution of maximum resource per node, one layer per type of building
BuildingMatrix2 = zeros(dim,dim,Numberofbuildings, TotalTimesteps);                   %↙
Dummy Matrix, for constructing maximum resources per node, per building
ResourceMatrix = zeros(dim,dim,amountofcivilians,Numberofdifferentbuildings,↙
TotalTimesteps);        %live resource value
ResourceInfoMatrix = zeros(dim,dim,amountofcivilians, 2*Numberofdifferentbuildings+1,↙
TotalTimesteps+1);  %Information gathered from ResourceMatrix for every civilian
CivilianInfoMatrix = zeros(dim,dim, amountofcivilians, CivilianTagVariables+5,↙
TotalTimesteps);          %matrix for storage of information of Civilians
CivilianMemoryMatrix = zeros(dim,dim, amountofcivilians, CivilianMemoryVariables,↙
TotalTimesteps); %matrix for storage of the memory of Civilians, e.g. the minimum and↙
maximum fear
FearMatrix = zeros(dim,dim,TotalTimesteps); %fear scent on every node, 0 at start
TerroristInfoMatrix = zeros(dim,dim,amountofcivilians,TerroristInfoVariables,↙
TotalTimesteps);
attacks = zeros(TotalTimesteps,1); %Stores the number of attacks per timestep
NumberofSecurityAgents = zeros(TotalTimesteps,1); %Stores the number of security↙
agents per timestep
arrests = zeros(TotalTimesteps,1); %stores the number of arrests per timestep
AnalysisMatrix = zeros(3,TotalTimesteps); %stores information on attacks per timestep
Timecounter = zeros(TotalTimesteps,1);
%%% Create Environment
CreateBuilding(40,10,30,1); %Place in picture of Bulleit: 1a
CreateBuilding(25,5,30,2); %Place in picture of Bulleit: 3a
CreateBuilding(15,15,30,3); %Place in picture of Bulleit: 4a
CreateBuilding(5,25,30,4); %Place in picture of Bulleit: 2b
CreateBuilding(25,25,30,5); %Place in picture of Bulleit: 3b
CreateBuilding(45,25,30,6); %Place in picture of Bulleit: 1b
CreateBuilding(10,40,30,7); %Place in picture of Bulleit: 2a
CreateBuilding(25,45,30,8); %Place in picture of Bulleit: 3c
CreateBuilding(35,35,30,9); %Place in picture of Bulleit: 4b
BuildingMatrix(:,:,1,1) = BuildingMatrix2(:,:,1,1) + BuildingMatrix2(:,:,6,1); %Making↙
sure that all buildings of the same type are in 1 layer
BuildingMatrix(:,:,2,1) = BuildingMatrix2(:,:,7,1) + BuildingMatrix2(:,:,4,1);
BuildingMatrix(:,:,3,1) = BuildingMatrix2(:,:,2,1) + BuildingMatrix2(:,:,5,1) +↙
BuildingMatrix2(:,:,8,1);
BuildingMatrix(:,:,4,1) = BuildingMatrix2(:,:,3,1) + BuildingMatrix2(:,:,9,1);
%%% Create Initial situation
CreateCivilians(Maxcurve, MaxSlope);
CreateCivilianInfo(CivilianTagVariables,Numberofdifferentbuildings);
CreateResources(Numberofdifferentbuildings);
CreateResourceInfo(Numberofdifferentbuildings);
CreateCivilianMemory(CivilianTagVariables,Numberofdifferentbuildings,1);
Timecounter(1,1)=1;
%%% Running program for other timesteps
for AmountTimesteps = 1:TotalGroupTimesteps
    for T = 2:TotalTimesteps
        FearSpread(T);
        CivilianWalking(amountofcivilians,CivilianTagVariables,↙
Numberofdifferentbuildings, T);
        GrowbackBuilding(Numberofdifferentbuildings,T);
        TerroristWalking(amountofcivilians,CivilianTagVariables,↙
Numberofdifferentbuildings, T);
        CivilianInteraction(CivilianTagVariables,Pb,Pic,T);
        BecomesTerrorist(CivilianTagVariables,T);
        BecomeActiveTerrorist(CivilianTagVariables,Numberofdifferentbuildings,T);
        RemainactiveTerrorist(Numberofdifferentbuildings,T);
        SurveillanceData(Numberofdifferentbuildings, T);
        Attack(Numberofdifferentbuildings, T);
        SecurityWalking(T);
        SecurityAgents(Numberofdifferentbuildings,T);
        SecurityCheck(CivilianTagVariables,T);
        AgentsDie(Numberofdifferentbuildings,T);
        if AmountTimesteps == 1
            GompertzTime = (AmountTimesteps-1)*TotalTimesteps+T;
        else
            GompertzTime = (AmountTimesteps-1)*TotalTimesteps+T-(AmountTimesteps-1);
        end
        AgentsCreated(Maxcurve, MaxSlope,Numberofdifferentbuildings,↙
CivilianTagVariables,GompertzTime,T);
        if AmountTimesteps == 1
            (AmountTimesteps-1)*TotalTimesteps+T
            Timecounter(T,1) = (AmountTimesteps-1)*TotalTimesteps+T;
        else
            (AmountTimesteps-1)*TotalTimesteps+T-(AmountTimesteps-1)
            Timecounter(T,1) = (AmountTimesteps-1)*TotalTimesteps+T-(AmountTimesteps-↙
1);
        end
        if T==TotalTimesteps
            SaveMatrices(versionnumber,AmountTimesteps,T);
        end
    end
end
```

```matlab
%%Functions used (indirectly) by the above code

function AddSecurityAgents(plussecagents, Timestep)
%AddSecurityAgents   Function to add Security agents randomly taking
%historical nodal wealth into account
%AddSecurityAgents(plussecagents, Timestep) adds a number of plussecagents
%on time=Timestep to the area as Security
global AreaMatrix
global ResourceMatrix
Size = size(AreaMatrix);
NodalWealth = zeros(Size(1),Size(2));
%Calculating nodal wealth
for i = 1:Size(1)
    for j = 1:Size(2)
        NodalWealth(i,j) = sum(sum(ResourceMatrix(i,j,:,:,Timestep)));
    end
end
TotalWealth = sum(sum(NodalWealth));
X = randi([0,int32(TotalWealth)],Size(1),Size(2));
%Trying to add security agents on nodes with high nodal wealth
for i = 1:Size(1)
    for j = 1:Size(2)
        if plussecagents > 0
            if NodalWealth(i,j) > X(i,j)
                k = 1;
                while AreaMatrix(i,j,k,Timestep)~=0
                    k = k+1;
                end
                AreaMatrix(i,j,k,Timestep) = 4;
                plussecagents = plussecagents-1;
            end
        end
    end
end
%if we have still not added all needed security agents, then we add them
%randomly on i,j nodes where agents are
while plussecagents>0
    nr = 0;
    for i = 1:Size(1)
        for j = 1:Size(2)
            if AreaMatrix(i,j,1)~=0
                nr = nr + 1;
            end
        end
    end
    x = randi([0,nr]);
    place1 = int32(Size(1)/2);
    place2 = int32(Size(2)/2);
    for i = 1:Size(1)
        for j = 1:Size(2)
            if AreaMatrix(i,j,1)~=0
                x = x-1;
                if x == 0
                    place1 = i;
                    place2 = j;
                end
            end
        end
    end
    k = 1;
    while AreaMatrix(place1,place2,k,Timestep)~=0
        k = k+1;
    end
    AreaMatrix(place1,place2,k,Timestep) = 4;
    plussecagents = plussecagents-1;
end
end

function [ACR] = AdjustedCriticalResource(i,j,k,resource,Numberofdifferentbuildings,↙
Timestep)
%AdjustedCriticalResource   Function that calculates the Adjusted Critical
%Resourcevalue for a given node and a given resource
%[ACR] = AdjustedCriticalResource(i,j,l,resource,Numberofdifferentbuildings,Timestep)
%calculates te Adjusted Critical Resource for the node (i,j,k) on the given↙
time=Timestep
%and returns this as value ACR
global BuildingMatrix
global CivilianMemoryMatrix
global ResourceInfoMatrix
global FearMatrix
a = BuildingMatrix(i,j,resource,Timestep);
b = CivilianMemoryMatrix(i,j,k,1,Timestep);
c = FearMatrix(i,j,Timestep);
d = CivilianMemoryMatrix(i,j,k,3,Timestep);
e = CivilianMemoryMatrix(i,j,k,2,Timestep);
f = ResourceInfoMatrix(i,j,k,2*Numberofdifferentbuildings+1,Timestep);
ACR = a - a*b*(1-exp(-(c-d)/(max(e-d,1))*(c-d)/(max(9*f,1))));
end

function AgentsCreated(Maxcurve, MaxSlope, Numberofdifferentbuildings,↙
CivilianTagVariables,GompertzTime,Timestep)
%AgentsCreated   Function to know which agents reproduce in order to
%create another Civilian Agent
%AgentsCreated(Maxcurve, MaxSlope, Numberofdifferentbuildings,CivilianTagVariables,↙
Timestep)
%Creates new agents based on the Gompertz function on the given time=Timestep
%See also Gompertz
global AreaMatrix
global CivilianInfoMatrix
global ResourceMatrix
global ResourceInfoMatrix
global CivilianMemoryMatrix
global BuildingMatrix
Size = size(AreaMatrix);
totalcuragents = 0;
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep)~=0 && AreaMatrix(i,j,k,Timestep)~=4
                totalcuragents = totalcuragents+1;
            end
        end
    end
end
theoreticaltotal  = int32(Gompertz(Maxcurve,MaxSlope,GompertzTime));
newagents = theoreticaltotal - totalcuragents;
if totalcuragents> theoreticaltotal
    newagents = 1;
end
places = randi([1,Size(1)],newagents,2);
```

```
for number = 1:newagents
    knew = 1;
    i = places(1);
    j = places(2);
    while AreaMatrix(i,j,knew,Timestep)~=0 && knew < Size(3)
        knew=knew+1;
    end
    AreaMatrix(i,j,knew,Timestep)=1;
    X = randi([0,9],1,CivilianTagVariables); %tags
    Y = randi([1,Numberofdifferentbuildings]); %which resource to find
    Z = randi([1,80]); %age
    W = randi([1,2]); %gender
    V = randi([1,5]); %vision
    U = randi([1,10]); % nervousness factor
    for a = 1:CivilianTagVariables
        CivilianInfoMatrix(i,j,knew,a,Timestep) = X(a);
    end
    CivilianInfoMatrix(i,j,knew,CivilianTagVariables+1,Timestep) = Y;
    CivilianInfoMatrix(i,j,knew,CivilianTagVariables+2,Timestep) = V;
    CivilianInfoMatrix(i,j,knew,CivilianTagVariables+3,Timestep) = W;
    CivilianInfoMatrix(i,j,knew,CivilianTagVariables+4,Timestep) = Z;
    CivilianInfoMatrix(i,j,knew,CivilianTagVariables+5,Timestep) = U;
    X = randi([100,300],1,Numberofdifferentbuildings)/10; %initial amount each↙
resource
    for a = 1:Numberofdifferentbuildings
        ResourceMatrix(i,j,knew,a,Timestep) = X(a);
    end
    X = randi([30,150],Numberofdifferentbuildings)/20; %metabolism
    for l = 1:Numberofdifferentbuildings
        ResourceInfoMatrix(i,j,knew,l,Timestep) = X(l);
        ResourceInfoMatrix(i,j,knew,Numberofdifferentbuildings+l,Timestep) =↙
ResourceMatrix(i,j,knew,l,Timestep)/ResourceInfoMatrix(i,j,knew,l,Timestep);
    end
    U = randi([1,99])/100;
    CivilianMemoryMatrix(i,j,knew,1,Timestep) = U; %innate nervousnes factor
    CivilianMemoryMatrix(i,j,knew,2,Timestep) = 0; %max fear
    CivilianMemoryMatrix(i,j,knew,3,Timestep) = 0; %min fear
    for m = 1:Numberofdifferentbuildings
        CivilianMemoryMatrix(i,j,knew,4+m,Timestep) = BuildingMatrix(i,j,m,Timestep);%↙
max resource
        CivilianMemoryMatrix(i,j,knew,3+Numberofdifferentbuildings+1+m,Timestep) = i;
        CivilianMemoryMatrix(i,j,knew,3+2*Numberofdifferentbuildings+1+m,Timestep) =↙
j;
    end
end
end

function AgentsDie(Numberofdifferentbuildings,Timestep)
%AgentsDie   Function to determine which agents are not able to live another timestep↙
(gw<1)
%AgentsDie(Numberofdifferentbuildings,Timestep) removes the agents of which
%the generalized wealth is below 1.0 from the area for the given
%time=Timestep
%See also GeneralizedWealth
global AreaMatrix
global ResourceInfoMatrix
Size = size(AreaMatrix);
for i =1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
```

```
            if AreaMatrix(i,j,k,Timestep)~=0
                if ResourceInfoMatrix(i,j,k,2*Numberofdifferentbuildings+1,Timestep) <↙
1
                    AreaMatrix(i,j,k,Timestep)=0;
                end
            end
        end
    end
end

end

function Attack(Numberofdifferentbuildings,Timestep)
%Attack   Function to check whether an active Terrorist Agent does an attack, and
%performes all necessary consequences
%Attack(Numberofdifferentbuildings,Timestep) checks for the given time=Timestep
%whether active terrorist agents are doing an attack and stores information
%of these attacks
global AreaMatrix
global TerroristInfoMatrix
global ResourceInfoMatrix
global BuildingMatrix
global FearMatrix
global attacks
global AnalysisMatrix
Size = size(AreaMatrix);
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep) == 3 %check for active agents
                mu = TerroristInfoMatrix(i,j,k,6,Timestep);
                sigma = TerroristInfoMatrix(i,j,k,7,Timestep);
                if TerroristInfoMatrix(i,j,k,8,Timestep)>mu + sigma && mu/sigma>0.25↙
&& sum(TerroristInfoMatrix(i,j,k,1:5,Timestep)~=0)==5 %conditions for performing↙
attack
                    attacks(Timestep) = attacks(Timestep) + 1;
                    fear = 0;
                    %calculation of total fear in surrounding area, based
                    %on attack
                    for m = -1:1
                        for n = -1:1
                            if i+m>0 && j+n >0 && i+m <Size(1)+1 && j+n < Size(2)+1
                                fear = fear + ResourceInfoMatrix(i+m,j+n,k,↙
2*Numberofdifferentbuildings+1,Timestep);
                            end
                        end
                    end
                    for m = -1:1%updating all information in surrounding area
                        for n = -1:1
                            if i+m>0 && j+n >0 && i+m <Size(1)+1 && j+n < Size(2)+1
                                AnalysisMatrix(1,Timestep) = AnalysisMatrix(1,↙
Timestep) + sum(sum(ResourceInfoMatrix(i+m,j+n,:,:,Timestep)));
                                ResourceInfoMatrix(i+m,j+n,:,:,Timestep)=0; %all↙
resources destroyed
                                AnalysisMatrix(2,Timestep) = AnalysisMatrix(2,↙
Timestep) + sum(BuildingMatrix(i+m,j+n,:,Timestep));
                                BuildingMatrix(i+m,j+n,:,Timestep)=0; %all buildings↙
destroyed
                                FearMatrix(i+m,j+n,Timestep) = fear; %fear created
                                AnalysisMatrix(3,Timestep) = AnalysisMatrix(3,↙
```

```
Timestep) + sum(AreaMatrix(i+m,j+n,:,Timestep));
                                AreaMatrix(i+m,j+n,:,Timestep)=0; %all agents↙
destroyed
                            end
                        end
                    end
                end
            end
        end
    end
end
end

function BecomeActiveTerrorist(CivilianTagVariables,Numberofdifferentbuildings,↙
Timestep)
%BecomeActiveTerrorist   Function to determine whether an inactive
%Terrorist becomes an active terrorist
%BecomeActiveTerrorist(CivilianTagVariables,Numberofdifferentbuildings,Timestep)
%Checks for the given time=Timestep whether inactive terrorists become
%active based on the Generalized Wealth and age
%See also GeneralizedWealth
global AreaMatrix
global ResourceInfoMatrix
global CivilianInfoMatrix
Size = size(AreaMatrix);
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep) == 2
                gw = ResourceInfoMatrix(i,j,k,2*Numberofdifferentbuildings+1,↙
Timestep);
                if CivilianInfoMatrix(i,j,k,CivilianTagVariables+4,Timestep) > 18 &&↙
gw>5.0 %18 && 5.0 (note)
                    AreaMatrix(i,j,k,Timestep) = 3;
                end
            end
        end
    end
end
end

function BecomesTerrorist(CivilianTagVariables,Timestep)
%BecomesTerrorist   Function to determine whether or not a Civilian Agent
%becomes a Terrorist Agent, based on Tagvalues
%BecomesTerrorist(CivilianTagVariables,Timestep) checks all agents whether
%they become inactive terrorists, based on their cultural identity
global AreaMatrix
global CivilianInfoMatrix
Size = size(AreaMatrix);
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep) == 2 %Also Terrorist Agent have every↙
timestep the same probability to stay Terrorist
                AreaMatrix(i,j,k,Timestep) = 1;
            end
            if AreaMatrix(i,j,k,Timestep) == 1
                culturalidentity = 0;
                for m = 1:CivilianTagVariables %calculating total cultural identity↙
(=Total tagvalue)
```

```
                    culturalidentity = culturalidentity+CivilianInfoMatrix(i,j,k,m,↙
Timestep);
                end
                P = abs(((culturalidentity - 22.5)/22.5)^5);
                x = randi([0,100])/100;
                if x<P
                    AreaMatrix(i,j,k,Timestep) = 2;
                end
            end
        end
    end
end
end

function [m, n] = CheckSecuritySurroundings(i,j,m,n,nr,Timestep)
%CheckSecuritySurroundings   function to check the surroundings of a Security Agent
%[m, n] = CheckSecuritySurroundings(i,j,m,n,nr,Timestep) checks the
%surrounding of a security agents and returns a place [m,n] on which an
%agent other than a security agent is situated
global AreaMatrix
Size = size(AreaMatrix);
if nr == 100
    m = 0;
    n = 0;
else
    nr = nr + 1;
    if i+m==0 || j+n==0 || j+n>Size(2) || i+m>Size(1)
        m = randi([-1,1]);
        n = randi([-1,1]);
        [m,n] = CheckSecuritySurroundings(i,j,m,n,nr,Timestep);
    elseif AreaMatrix(i+m,j+n,1,Timestep) == 0 || AreaMatrix(i+m,j+n,1,Timestep) == 4
        m = randi([-1,1]);
        n = randi([-1,1]);
        [m,n] = CheckSecuritySurroundings(i,j,m,n,nr,Timestep);
    end
end
end

function CivilianInteraction(CivilianInfoVariables,Pb,Pic,Timestep)
%CivilianInteraction   Function to go over all non-security Agents and let them↙
interact their
%Tags with one of the Agents around them.
%CivilianInteraction(CivilianInfoVariables,Pb,Pic,Timestep)
global AreaMatrix
global CivilianInfoMatrix
Size = size(AreaMatrix);
X = randi([-1,1],Size(1),Size(2),Size(3),2);
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep) == 1 || AreaMatrix(i,j,k,Timestep) == 2 ||↙
AreaMatrix(i,j,k,Timestep) == 3
                a = X(i,j,k,1); %randomly deciding which surrounding node we have to↙
choose
                b = X(i,j,k,2);
                if a+i<1 || a+i>Size(1) ||j+b<1 || j+b>Size(2) %check and update↙
if/when the new index is out of bounds
                    while a+i<1 || a+i>Size(1) ||j+b<1 || j+b>Size(2)
                        a = randi([-1,1]);
                        b = randi([-1,1]);
```

```
                    end
                end
                if AreaMatrix(i+a,j+b,1,Timestep)==1 || AreaMatrix(i,j,k,Timestep) ==↙
2 || AreaMatrix(i,j,k,Timestep) == 3%do not interact with Security Agents
                    difference = DifferenceTagsCivilians(i,j,k,i+a,j+b,1,↙
CivilianInfoVariables,Timestep); %calculate difference in Tagvalue
                    g = randi([0,100])/100;
                    if g<(1-difference/45)
                        Tag = randi([1,CivilianInfoVariables],1,2);
                        if CivilianInfoMatrix(i,j,k,Tag(1),Timestep) ~=↙
CivilianInfoMatrix(i+a,j+b,1,Tag(1),Timestep)
                            c = randi([1,2]); %determine what to do
                            if c == 1 %agent that moves changes Tag to match
                                CivilianInfoMatrix(i,j,k,Tag(1),Timestep) =↙
CivilianInfoMatrix(i+a,j+b,1,Tag(1),Timestep);
                            elseif c == 2 %agent that moves has a potential radical↙
change
                                Prc = Pb*difference/9;
                                d = randi([0,100],1,2)/100;
                                if d(1)<Prc %Determine whether or not a radical change↙
happes
                                    if d(2)<CivilianInfoMatrix(i,j,k,Tag(1),Timestep)↙
/9
                                        CivilianInfoMatrix(i,j,k,Tag(1),Timestep) = 9;
                                    else
                                        CivilianInfoMatrix(i,j,k,Tag(1),Timestep) = 0;
                                    end
                                end
                            end
                        end
                        e = randi([0,100])/100;
                        if e<Pic %The isolated probability to have a +-1 change of↙
another random Tagvalue
                            f = randi([0,1]);
                            if f == 0
                                f = -1;
                            end
                            CivilianInfoMatrix(i,j,k,Tag(2),Timestep) =↙
CivilianInfoMatrix(i,j,k,Tag(2),Timestep) + f;
                        end
                    end
                end
            end
        end
    end
end
end

function [place] = CivilianOptimalWalk(i,j,k,maxplace1,maxplace2,CivilianTagVariables,↙
Numberofdifferentbuildings, Timestep)
%CivilianOptimalWalk   Function to search for the place the Civilian wants to go to,↙
taking into
%account the place where the maximum encountered resourcevalue is
%[place] = CivilianOptimalWalk(i,j,k,maxplace1,maxplace2,CivilianTagVariables,↙
Numberofdifferentbuildings, Timestep)
%See also AdjustedCriticalResource

global AreaMatrix
global CivilianInfoMatrix
Size = size(AreaMatrix);
```

```
z1 = 0;
z2 = 0;
z3 = -1;
search = CivilianInfoMatrix(i,j,k,CivilianTagVariables+1,Timestep); %get the number of↙
the to harvest resource
%Going over all surrounding nodes
for m = -1:1
    for n = -1:1
        if i+m>0 && j+n>0 && j+n<Size(2)+1 && i+m<Size(1)+1 && (m~=0 || n~=0) %Check↙
that the Civilian is not going out of bounds
            if abs(i+m-maxplace1)<abs(i-maxplace1)+1 || abs(j+n-maxplace2) < abs(j-↙
maxplace2)+1 %Check that the Civilian is going into the direction of the maximum↙
encountered resourcevalue
                ACR = AdjustedCriticalResource(i+m,j+n,k,search,↙
Numberofdifferentbuildings,Timestep); %Adjusted critical resourcevalue for given node↙
and resource
                if ACR>=z3 %updating newplace
                    z1 = i+m;
                    z2 = j+n;
                    z3 = ACR;
                end
            end
        end
    end
end
place = [z1, z2];
end

function CivilianWalking(amountofcivilians,CivilianTagVariables,↙
Numberofdifferentbuildings, Timestep)
%CivilianWalking  Function to move every Civilian Agent from their old place on↙
Timestep-1 to a new place
%on Timestep taking into account the most optimal walk
%CivilianWalking(amountofcivilians,CivilianTagVariables, Numberofdifferentbuildings,↙
Timestep)
%See also CivilianOptimalWalk, ReplaceInfo
global AreaMatrix
global BuildingMatrix
global CivilianMemoryMatrix
global CivilianInfoMatrix
Size = size(AreaMatrix);
BuildingMatrix(:,:,:,Timestep) = BuildingMatrix(:,:,:,Timestep-1); %Update of all↙
buildingresource values
%running over all matrix nodes
for i = 1:Size(1)
    for j = 1:Size(2)
        for k=1:amountofcivilians
            if AreaMatrix(i,j,k,Timestep-1) == 1 %check whether or not a civilian is↙
present at node
                l = CivilianInfoMatrix(i,j,k,CivilianTagVariables+1,Timestep-1); %↙
resource to harves
                maxplace1 = CivilianMemoryMatrix(i,j,k,1+Numberofdifferentbuildings,↙
Timestep-1); %node of maximum encountered value for the resource that needs to be↙
harvested
                maxplace2 = CivilianMemoryMatrix(i,j,k,1+2*Numberofdifferentbuildings,↙
Timestep-1);
                newplace = CivilianOptimalWalk(i,j,k,maxplace1,maxplace2,↙
CivilianTagVariables,Numberofdifferentbuildings, Timestep-1); %place where the↙
civilian wants to go to on Timestep
                if newplace(1)>0 && newplace(2)>0
```

```
                    ReplaceInfo(i,j,k,newplace(1),newplace(2),l,↙
Numberofdifferentbuildings,CivilianTagVariables,Timestep); %Move all needed↙
information to the new place
                end
            end
        end
    end
end
end

function CreateBuilding(i,j,resourcefactor, BuildingNumber)
%CreateBuilding   Creates a Building on Matrixpoint i,j with a resourcefactor↙
distributed
%around the Matrix with Von Neumann Neighbourhood and maximum
%resourcefactor 30.0, minimumresourcefactor 1.0 (steps of 1.0)
%CreateBuilding(i,j,resourcefactor, BuildingNumber)
global BuildingMatrix2
Size = size(BuildingMatrix2);
BuildingMatrix2(i,j,BuildingNumber) = resourcefactor;
for k = 1:(2*max(Size(1),Size(2)))
    for m = 0:k
        if k < resourcefactor+1
            if (i+k-m)<(Size(1)+1) && (j+m) < (Size(2)+1)
                BuildingMatrix2(i+k-m,j+m,BuildingNumber) = resourcefactor-k+1;
            end
            if (i-k+m)>0 && (j-m) > 0
                BuildingMatrix2(i-k+m,j-m,BuildingNumber) = resourcefactor-k+1;
            end
            if (i-m)>0 && (j+k-m) < (Size(2)+1)
                BuildingMatrix2(i-m,j+k-m,BuildingNumber) = resourcefactor-k+1;
            end
            if (i+m)<(Size(1)+1) && (j-k+m) > 0
                BuildingMatrix2(i+m,j-k+m,BuildingNumber) =  resourcefactor-k+1;
            end
        end
    end
end
end

function CreateCivilianInfo(CivilianTagVariables,Numberofbuildings)
%CreateCivilianInfo   Creates all necessary information for every Civilian on time=1
CreateCivilianInfo(CivilianTagVariables,Numberofbuildings)
%for third matrix component of CivilianInfoMatrix the following holds:
%1: tag1, 2:tag2, 3:tag3, 4:tag4, 5:tag5, 6:wich resource to find, 7:
%vision , 8: gender, 9: age, 10: (initial) nervousness factor
global AreaMatrix
global CivilianInfoMatrix
Size = size(AreaMatrix);
X = randi([0,9],Size(1),Size(2),CivilianTagVariables); %tags
Y = randi([1,Numberofbuildings],Size(1),Size(2)); %which resource to find
Z = randi([1,80],Size(1),Size(2)); %age
W = randi([1,2], Size(1),Size(2)); %gender
V = randi([1,5],Size(1),Size(2)); %vision
U = randi([1,10],Size(1),Size(2)); % nervousness factor
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:CivilianTagVariables
            if AreaMatrix(i,j,1,1) == 1
                CivilianInfoMatrix(i,j,1,k,1) = X(i,j,k);
            end
```

```
            end
        end
    end
for i = 1:Size(1)
    for j = 1:Size(2)
        if AreaMatrix(i,j,1,1) == 1
            CivilianInfoMatrix(i,j,1,CivilianTagVariables+1,1) = Y(i,j);
            CivilianInfoMatrix(i,j,1,CivilianTagVariables+2,1) = V(i,j);
            CivilianInfoMatrix(i,j,1,CivilianTagVariables+3,1) = W(i,j);
            CivilianInfoMatrix(i,j,1,CivilianTagVariables+4,1) = Z(i,j);
            CivilianInfoMatrix(i,j,1,CivilianTagVariables+5,1) = U(i,j);
        end
    end
end
end

function CreateCivilianMemory(CivilianTagVariables,Numberofdifferentbuildings,↙
Timestep)
%CreateCivilianMemory   Stores Maximum and minimum fear encountered + maximum value↙
found per resource
%taking into account the vision the agent has
%CreateCivilianMemory(CivilianTagVariables,Numberofdifferentbuildings,Timestep)
global AreaMatrix
global BuildingMatrix
global CivilianMemoryMatrix
global CivilianInfoMatrix
Size = size(AreaMatrix);
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep) == 1
                U = randi([1,99])/100;
                CivilianMemoryMatrix(i,j,k,1,Timestep) = U; %innate nervousnes factor
                CivilianMemoryMatrix(i,j,k,2,Timestep) = 0; %max fear
                CivilianMemoryMatrix(i,j,k,3,Timestep) = 0; %min fear
                for m = 1:Numberofdifferentbuildings
                    vision = CivilianInfoMatrix(i,j,k,CivilianTagVariables+2,↙
Timestep);
                    maxres = -1;
                    for n1 = -vision:vision
                        for n2 = -vision:vision
                            if i+n1>0 && j+n2>0 && j+n2<Size(2)+1 && i+n2<Size(1)+1 &&↙
(n2~=0 || n2~=0)
                                if (n1 == 0 || n2 == 0) && (n1~=0 || n2~=0)
                                    if  BuildingMatrix(i+n1,j+n2,m,Timestep)>maxres;
                                        maxres = BuildingMatrix(i+n1,j+n2,m,Timestep);
                                        place1 = i+n1;
                                        place2 = j+n2;
                                    end
                                end
                            end
                        end
                    end
                    CivilianMemoryMatrix(i,j,k,4+m,Timestep) = maxres;%max resource
                    CivilianMemoryMatrix(i,j,k,4+Numberofdifferentbuildings+m,↙
Timestep) = place1;
                    CivilianMemoryMatrix(i,j,k,4+2*Numberofdifferentbuildings+m,↙
Timestep) = place2;
                end
            end
```

```
        end
    end
end
end

function CreateCivilians(Maxcurve, MaxSlope)
%CreateCivilians   Creates a pregiven expected amount (based on the Gompertz function)
%of civilians in the grid of the Matrix (randomly distributed)
%CreateCivilians(Maxcurve, MaxSlope)
%See also Gompertz
global AreaMatrix
amount = Gompertz(Maxcurve,MaxSlope,1);
Size = size(AreaMatrix);
randomlijst = rand(Size(1),Size(2));
chance = amount/(Size(1)*Size(2));
for i = 1:Size(1)
    for j = 1:Size(2)
        if sum(sum(AreaMatrix(:,:,1,1)))<int32(amount)+1
            if randomlijst(i,j)<chance
                AreaMatrix(i,j,1,1) = 1;
            end
        end
    end
end
end

function CreateResourceInfo(Numberofdifferentbuildings)
%CreateResourceInfo   Creating initial values of metabolism for resources for every
%agent and calculating the first generalized resource/wealth
%CreateResourceInfo(Numberofdifferentbuildings)
%for 4th entry of ResourceInfoMatrix:: 1-Numberofdifferentbuildings:
%metabolism, Numberofdifferentbuildings+1-2*Numberofdifferentbuildings:
%generalized resource, 2*Numberofdifferentbuildings+1: Generalized wealth
global AreaMatrix
global ResourceMatrix
global ResourceInfoMatrix
Size = size(AreaMatrix);
X = randi([30,150],Size(1),Size(2),Numberofdifferentbuildings)/20; %metabolism resources for every agent
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Numberofdifferentbuildings
            if AreaMatrix(i,j,1,1) == 1
                ResourceInfoMatrix(i,j,1,k,1) = X(i,j,k);
                ResourceInfoMatrix(i,j,1,Numberofdifferentbuildings+k,1) = ResourceMatrix(i,j,1,k,1)/ResourceInfoMatrix(i,j,1,k,1);
            end
        end
    end
end
list = zeros(Size(1),Size(2),1);
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Numberofdifferentbuildings
            list(i,j,k) = ResourceInfoMatrix(i,j,1,Numberofdifferentbuildings+k,1);
        end
    end
end
for i = 1:Size(1)
    for j = 1:Size(2)
        ResourceInfoMatrix(i,j,1,2*Numberofdifferentbuildings+1,1) = min(list(i,j));
    end
end
end

function CreateResources(Numberofdifferentbuildings)
%CreateResources   Creating initial values of amounts for resources for every agent
%CreateResources(Numberofdifferentbuildings)
global AreaMatrix
global ResourceMatrix
%for 4th entry of ResourceMatrix:: 1-Numberofdifferentbuildings: amount
Size = size(AreaMatrix);
X = randi([100,300],Size(1),Size(2),Numberofdifferentbuildings)/10; %initial amount each resource per agent

for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Numberofdifferentbuildings
            if AreaMatrix(i,j,1,1) == 1
                ResourceMatrix(i,j,1,k,1) = X(i,j,k);
            end
        end
    end
end
end

function [difference] = DifferenceTagsCivilians(i,j,k,i2,j2,k2,CivilianInfoVariables,Timestep)
%DifferenceTagsCivilians   Function to determine the differene in total tagvalue
%between 2 agents (agent (i,j,k) and agent (i2,j2,k2)), this value is returned as [difference]
%[difference] = DifferenceTagsCivilians(i,j,k,i2,j2,k2,CivilianInfoVariables,Timestep)
global CivilianInfoMatrix
difference = 0;
for Tag = 1:CivilianInfoVariables
    difference = difference + abs(CivilianInfoMatrix(i,j,k,Tag,Timestep) - CivilianInfoMatrix(i2,j2,k2,Tag,Timestep));
end
end

function FearSpread(Timestep)
%FearSpread   Function to spread the created fear across the environment
%FearSpread(Timestep)
global AreaMatrix
global FearMatrix
Size = size(AreaMatrix);
for i = 1:Size(1)
    for j = 1:Size(2)
        fear = 0;
        nr = 0;
        for m = -1:1
            for n = -1:1
                if i+m>0 && j+n>0 && j+n<Size(2)+1 && i+m<Size(1)+1
                    fear = fear + FearMatrix(i+m,j+n,Timestep-1);
                    nr = nr + 1;
                end
            end
        end
        FearMatrix(i,j,Timestep) = fear/nr;
    end
end
end

function GeneralizedResource(inew,jnew,knew,Numberofdifferentbuildings,Timestep)
%GeneralizedResource   Function to determine all Generalized Resourcevalues for a given Civilian
%or Terrorist Agent (inew, jnew, knew)
%GeneralizedResource(inew,jnew,knew,Numberofdifferentbuildings,Timestep)
global ResourceMatrix
global ResourceInfoMatrix
for m = 1:Numberofdifferentbuildings
    ResourceInfoMatrix(inew,jnew,knew,Numberofdifferentbuildings+m,Timestep) = ResourceMatrix(inew,jnew,knew,m,Timestep)/max(ResourceInfoMatrix(inew,jnew,knew,m,Timestep),1);
end
end

function GeneralizedWealth(inew,jnew,knew,Numberofdifferentbuildings, Timestep)
%GeneralizedWealth   Function to determine the Generalized Wealth for a
%given Agent (inew, jnew, knew)
%GeneralizedWealth(inew,jnew,knew,Numberofdifferentbuildings, Timestep)
global ResourceInfoMatrix
dummy = [];
for m = Numberofdifferentbuildings+1:2*Numberofdifferentbuildings
    dummy(m-Numberofdifferentbuildings) = ResourceInfoMatrix(inew,jnew,knew,m,Timestep);
end
ResourceInfoMatrix(inew,jnew,knew,2*Numberofdifferentbuildings+1,Timestep) = min(dummy);
end

function [y] = Gompertz(Maxcurve,MaxSlope, Timestep)
%Gompertz   Function that calculates the population number for a given
%time=Timestep based on the Gompertz function and returns this value as [y]
%[y] = Gompertz(Maxcurve,MaxSlope,Timestep)
%For more information, see https://cran.r-project.org/web/packages/grofit/grofit.pdf
% 0.23\% growth rate in EU https://www.cia.gov/library/publications/the-world-factbook/geos/ee.html
% pop growth projections 0.15 https://esa.un.org/unpd/wpp/Publications/Files/Key_Findings_WPP_2015.pdf
% https://cran.r-project.org/web/packages/grofit/grofit.pdf (note)
y = Maxcurve*exp(-exp(MaxSlope*exp(1)/Maxcurve*(-Timestep-500)+1));
end

function GrowbackBuilding(Numberofdifferentbuildings,Timestep)
%GrowbackBuilding   Function to make sure that the values for the resources of the buildings
%grows back with 1/4 of the original value, but should not exceed the
%original value
%GrowbackBuilding(Numberofdifferentbuildings,Timestep)
%See also CreateBuilding
global BuildingMatrix
Size = size(BuildingMatrix);
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Numberofdifferentbuildings
            BuildingMatrix(i,j,k,Timestep) = max(0,BuildingMatrix(i,j,k,Timestep)+BuildingMatrix(i,j,k,1)*0.25);
            if BuildingMatrix(i,j,k,Timestep) > BuildingMatrix(i,j,k,1)
                BuildingMatrix(i,j,k,Timestep) = BuildingMatrix(i,j,k,1);
            end
        end
    end
end

end

function RemainactiveTerrorist(Numberofdifferentbuildings,Timestep)
%RemainactiveTerrorist   Function to check whether an active Terorrist Agent should stay active. If
%it becomes inactive all data is lost. This decision is based on the
%Generalized Wealth
%RemainactiveTerrorist(Numberofdifferentbuildings,Timestep)
%See also GeneralizedWealth
global AreaMatrix
global ResourceInfoMatrix
global TerroristInfoMatrix
Size = size(AreaMatrix);
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep) == 3
                gw = ResourceInfoMatrix(i,j,k,2*Numberofdifferentbuildings+1,Timestep);
                if gw<3.0 %3.0 (note) %Becomes inactive
                    AreaMatrix(i,j,k,Timestep) = 2;
                    TerroristInfoMatrix(i,j,k,1:7,Timestep)=0;
                end
            end
        end
    end
end
end

function RemoveSecurityAgents(minussecagents, Timestep)
%RemoveSecurityAgents   Function to remove Security agents randomly taking historical nodal wealth
%into account
%RemoveSecurityAgents(minussecagents, Timestep)
global AreaMatrix
global ResourceMatrix
Size = size(AreaMatrix);
HistoricalNodalWealth = zeros(Size(1),Size(2));
%Calculating historical nodal wealth
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep) == 4
                HistoricalTime = Timestep;
                if HistoricalTime-5<1
                    HistoricalTime = Timestep-1;
                end
                for T = 1:HistoricalTime
                    HistoricalNodalWealth(i,j) = HistoricalNodalWealth(i,j) + sum(sum(ResourceMatrix(i,j,:,:,HistoricalTime)));
                end
            end
        end
    end
end
```

```
TotalHistoricalWealth = sum(sum(HistoricalNodalWealth));
X = randi([0,int32(TotalHistoricalWealth)],Size(1),Size(2));
%Trying to remove security agents on nodes with high historical nodal wealth
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if minussecagents > 0
                if AreaMatrix(i,j,k,Timestep) == 4
                    if X(i,j) >  HistoricalNodalWealth(i,j)
                        AreaMatrix(i,j,k,Timestep) = 0;
                        minussecagents = minussecagents -1;
                    end
                end
            end
        end
    end
end
%if we have still not removed all needed to remove security agents, then we remove↙
them
%randomly on i,j nodes where agents are
while minussecagents>0
    place1 = 0;
    place2 = 0;
    place3 = 0;
    nr = 0;
    for i = 1:Size(1)
        for j = 1:Size(2)
            for k = 1:Size(3)
                if AreaMatrix(i,j,1)==4
                    nr = nr + 1;
                end
            end
        end
    end
    x = randi([0,nr]);
    for i = 1:Size(1)
        for j = 1:Size(2)
            for k = 1:Size(3)
                if AreaMatrix(i,j,k)==4
                    x = x-1;
                    if x == 0
                        place1 = i;
                        place2 = j;
                        place3 = k;
                    end
                end
            end
        end
    end
    if place1>0 && place2>0 && place3>0
        AreaMatrix(place1,place2,place3,Timestep) = 0;
        minussecagents = minussecagents-1;
    else
        minussecagents = 0;
    end
end
end

function ReplaceInfo(i,j,k, inew, jnew, harvest,Numberofdifferentbuildings,↙
CivilianTagVariables, Timestep)
%ReplaceInfo   Function to move all needed information from the old place i,j,k to the
%new place inew,jnew,knew on the new timestep Timestep.
%
%ReplaceInfo(i,j,k, inew, jnew, harvest,Numberofdifferentbuildings,↙
CivilianTagVariables, Timestep)
%
%See also GeneralizedResource, GeneralizedWealth
global AreaMatrix
global CivilianInfoMatrix
global BuildingMatrix
global ResourceMatrix
global ResourceInfoMatrix
global CivilianMemoryMatrix
global FearMatrix
Size = size(AreaMatrix);
knew=1;
while AreaMatrix(inew,jnew,knew,Timestep)~=0 %Looking for free place on node inew,jnew
    knew=knew+1;
end
%Area Update
AreaMatrix(inew,jnew,knew,Timestep) = AreaMatrix(i,j,k,Timestep-1);
%CivilianInfo general Update
CivilianInfoMatrix(inew,jnew, knew,:,Timestep) = CivilianInfoMatrix(i,j,k,:,Timestep-↙
1);
CivilianInfoMatrix(inew,jnew,knew,CivilianTagVariables+4,Timestep) =↙
CivilianInfoMatrix(inew,jnew,knew,CivilianTagVariables+4,Timestep) + 1/52; %(note)↙
1/52 is amount of time per timestep
%Resource Update
ResourceMatrix(inew,jnew,knew,:,Timestep) = ResourceMatrix(i,j,k,:,Timestep-1);
for h = 1:Numberofdifferentbuildings
    ResourceMatrix(inew,jnew,knew,h,Timestep) = ResourceMatrix(inew,jnew,knew,h,↙
Timestep) - ResourceInfoMatrix(i,j,1,h,1); %(note) should instead of -1 be the↙
metabolism?
end
ResourceMatrix(inew,jnew,knew, harvest, Timestep) = min(ResourceMatrix(i,j,k, harvest,↙
Timestep-1)+BuildingMatrix(inew,jnew, harvest, Timestep),30);
%ResourceInfo Update
ResourceInfoMatrix(inew,jnew,knew,:,Timestep) = ResourceInfoMatrix(i,j,k,:,Timestep-↙
1);
GeneralizedResource(inew,jnew,knew,Numberofdifferentbuildings,Timestep);
GeneralizedWealth(inew,jnew,knew,Numberofdifferentbuildings,Timestep);
BuildingMatrix(inew,jnew, harvest, Timestep) = 1; %Harvested all resources
%CivilianInfo Update
dummy = [];
for m = (Numberofdifferentbuildings+1):(2*Numberofdifferentbuildings)
dummy(m-(Numberofdifferentbuildings)) = ResourceInfoMatrix(inew,jnew,knew,m,Timestep);
end
placegw = find(dummy == ResourceInfoMatrix(inew,jnew,knew,↙
2*Numberofdifferentbuildings+1,Timestep));
CivilianInfoMatrix(inew,jnew,knew,CivilianTagVariables+1,Timestep) = placegw(1);
%CivilianMemory Update
CivilianMemoryMatrix(inew,jnew,knew,1,Timestep) = CivilianMemoryMatrix(i,j,k,1,↙
Timestep-1);  %innate nervousnes factor
CivilianMemoryMatrix(inew,jnew,knew,2,Timestep) = max(CivilianMemoryMatrix(i,j,k,2,↙
Timestep-1),FearMatrix(inew,jnew,Timestep)); %max fear
if CivilianMemoryMatrix(i,j,k,3,Timestep-1) ~=0
    CivilianMemoryMatrix(inew,jnew,knew,3,Timestep) = min(CivilianMemoryMatrix(i,j,k,↙
3,Timestep-1),FearMatrix(inew,jnew,Timestep)); %min fear
else
    CivilianMemoryMatrix(inew,jnew,knew,3,Timestep) = FearMatrix(inew,jnew,Timestep);↙
%min fear
end
for m = 1:Numberofdifferentbuildings
    vision = CivilianInfoMatrix(inew,jnew,knew,CivilianTagVariables+2,Timestep);
    maxres = -1;
    for n1 = -vision:vision
        for n2 = -vision:vision
            if i+n1>0 && j+n2>0 && j+n2<Size(2)+1 && i+n2<Size(1)+1 && (n2~=0 ||↙
n2~=0)
                if (n1 == 0 || n2 == 0) && (n1~=0 || n2~=0)
                    if  BuildingMatrix(i+n1,j+n2,m,Timestep)>maxres;
                        maxres = BuildingMatrix(i+n1,j+n2,m,Timestep);
                        place1 = i+n1;
                        place2 = j+n2;
                    end
                end
            end
        end
    end
    CivilianMemoryMatrix(inew,jnew,knew,4+m,Timestep) = max(CivilianMemoryMatrix(i,j,↙
k,4+m,Timestep-1),maxres);%max resource
    if CivilianMemoryMatrix(i,j,k,4+m,Timestep-1)<maxres
        CivilianMemoryMatrix(inew,jnew,knew,3+Numberofdifferentbuildings+1+m,Timestep)↙
= place1; %place i max resoure
        CivilianMemoryMatrix(inew,jnew,knew,3+2*Numberofdifferentbuildings+1+m,↙
Timestep) = place2; %place j max resource
    else
        CivilianMemoryMatrix(inew,jnew,knew,3+Numberofdifferentbuildings+1+m,Timestep)↙
= CivilianMemoryMatrix(i,j,k,3+Numberofdifferentbuildings+1+m,Timestep-1);
        CivilianMemoryMatrix(inew,jnew,knew,3+2*Numberofdifferentbuildings+1+m,↙
Timestep) = CivilianMemoryMatrix(i,j,k,3+2*Numberofdifferentbuildings+1+m,Timestep);
    end
end
end

function ReplaceTerroristInfo(i,j,k, inew, jnew,CivilianTagVariables,↙
Numberofdifferentbuildings, Timestep)
%ReplaceTerroristInfo   Function to move all needed information from the old place i,↙
j,k to the
%new place inew,jnew,knew on the new timestep Timestep.
%
%ReplaceTerroristInfo(i,j,k, inew, jnew,CivilianTagVariables,↙
Numberofdifferentbuildings, Timestep)
%
%See also GeneralizedResource, GeneralizedWealth
global AreaMatrix
global CivilianInfoMatrix
global ResourceMatrix
global BuildingMatrix
global TerroristInfoMatrix
global ResourceInfoMatrix
knew=1;
while AreaMatrix(inew,jnew,knew,Timestep)~=0
    knew=knew+1;
end
%Area Update
AreaMatrix(inew,jnew,knew,Timestep) = AreaMatrix(i,j,k,Timestep-1);
%CivilianInfo general Update
CivilianInfoMatrix(inew,jnew, knew,:,Timestep) = CivilianInfoMatrix(i,j,k,:,Timestep-↙
1);
CivilianInfoMatrix(inew,jnew,knew,CivilianTagVariables+4,Timestep) =↙
CivilianInfoMatrix(inew,jnew,knew,CivilianTagVariables+4,Timestep) + 1/52; %(note)↙
1/52 is amount of time per timestep
%Resource Update
ResourceMatrix(inew,jnew,knew,:,Timestep) = ResourceMatrix(i,j,k,:,Timestep-1);
for h = 1:Numberofdifferentbuildings
    ResourceMatrix(inew,jnew,knew,h,Timestep) = ResourceMatrix(inew,jnew,knew,h,↙
Timestep) - ResourceInfoMatrix(i,j,1,h,1); %(note) should instead of -1 be the↙
metabolism?
end
%ResourceInfo Update
ResourceInfoMatrix(inew,jnew,knew,:,Timestep) = ResourceInfoMatrix(i,j,k,:,Timestep-↙
1);
GeneralizedResource(inew,jnew,knew,Numberofdifferentbuildings,Timestep);
GeneralizedWealth(inew,jnew,knew,Numberofdifferentbuildings,Timestep);
%determine which resource to harvest
dummy = [];
for m = (Numberofdifferentbuildings+1):(2*Numberofdifferentbuildings)
dummy(m-Numberofdifferentbuildings) = ResourceInfoMatrix(inew,jnew,knew,m,Timestep);
end
harvest = find(dummy == ResourceInfoMatrix(inew,jnew,knew,↙
2*Numberofdifferentbuildings+1,Timestep));
CivilianInfoMatrix(inew,jnew,knew,CivilianTagVariables+1,Timestep) = harvest(1);
%Resource Update Harvest (cont.)
ResourceMatrix(inew,jnew,knew, harvest(1), Timestep) = min(ResourceMatrix(i,j,k,↙
harvest(1),Timestep-1)+BuildingMatrix(inew,jnew, harvest(1), Timestep),30);
BuildingMatrix(inew,jnew, harvest(1), Timestep) = 1; %Harvested all resources
%TerroristInfo update
TerroristInfoMatrix(inew,jnew,knew,:,Timestep) =  TerroristInfoMatrix(i,j,k,:,↙
Timestep-1);
end

function SaveMatrices(runnumber,AmountTimesteps,Timestep)
%SaveMatrices   Function to save Matrices locally
%
%SaveMatrices(runnumber,AmountTimesteps,Timestep)
global AreaMatrix
global BuildingMatrix
global ResourceMatrix
global ResourceInfoMatrix
global CivilianInfoMatrix
global CivilianMemoryMatrix
global FearMatrix
global TerroristInfoMatrix
global attacks
global NumberofSecurityAgents
global arrests
global AnalysisMatrix
global Timecounter
save(['Save-Timecounter-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'Timecounter', '-v7.3')
Timecounter(1,1)=Timecounter(Timestep,1);
Timecounter(2:end,1)=0;
save(['Save-AreaMatrix-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'AreaMatrix', '-v7.3')
AreaMatrix(:,:,:,1) = AreaMatrix(:,:,:,Timestep);
AreaMatrix(:,:,:,2:Timestep) = 0;
save(['Save-BuildingMatrix-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'BuildingMatrix', '-v7.3')
BuildingMatrix(:,:,:,1) = BuildingMatrix(:,:,:,Timestep);
```

```
BuildingMatrix(:,:,:,2:Timestep) = 0;
save(['Save-ResourceMatrix-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'ResourceMatrix', '-v7.3')
ResourceMatrix(:,:,:,:,1) = ResourceMatrix(:,:,:,:,Timestep);
ResourceMatrix(:,:,:,:,2:Timestep) = 0;
save(['Save-ResourceInfoMatrix-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'ResourceInfoMatrix', '-v7.3')
ResourceInfoMatrix(:,:,:,:,1) = ResourceInfoMatrix(:,:,:,:,Timestep);
ResourceInfoMatrix(:,:,:,:,2:Timestep) = 0;
save(['Save-CivilianInfoMatrix-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'CivilianInfoMatrix', '-v7.3')
CivilianInfoMatrix(:,:,:,:,1) = CivilianInfoMatrix(:,:,:,:,Timestep);
CivilianInfoMatrix(:,:,:,:,2:Timestep) = 0;
save(['Save-CivilianMemoryMatrix-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'CivilianMemoryMatrix', '-v7.3')
CivilianMemoryMatrix(:,:,:,:,1) = CivilianMemoryMatrix(:,:,:,:,Timestep);
CivilianMemoryMatrix(:,:,:,:,2:Timestep) = 0;
save(['Save-FearMatrix-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'FearMatrix', '-v7.3')
FearMatrix(:,:,1) = FearMatrix(:,:,Timestep);
FearMatrix(:,:,2:Timestep) = 0;
save(['Save-TerroristInfoMatrix-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'TerroristInfoMatrix', '-v7.3')
TerroristInfoMatrix(:,:,:,:,1) = TerroristInfoMatrix(:,:,:,:,Timestep);
TerroristInfoMatrix(:,:,:,:,2:Timestep) = 0;
save(['Save-attacks-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'attacks', '-v7.3')
attacks(1,1) = attacks(Timestep,1);
attacks(2:Timestep,1) = 0;
save(['Save-NumberofSecurityAgents-R' num2str(runnumber) '-' num2str(AmountTimesteps)↙
'.mat'],'NumberofSecurityAgents', '-v7.3')
NumberofSecurityAgents(1,1) = NumberofSecurityAgents(Timestep,1);
NumberofSecurityAgents(2:Timestep,1) = 0;
save(['Save-arrests-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'arrests', '-v7.3')
arrests(1,1) = arrests(Timestep,1);
arrests(2:Timestep,1) = 0;
save(['Save-AnalysisMatrix-R' num2str(runnumber) '-' num2str(AmountTimesteps) '.↙
mat'],'AnalysisMatrix', '-v7.3')
AnalysisMatrix(3,1) = AnalysisMatrix(3,Timestep);
AnalysisMatrix(3,2:Timestep) = 0;
end

function SecurityAgents(Numberofdifferentbuildings,Timestep)
%SecurityAgents   Function to decide where to add new Security agents or where to↙
remove
%security agents based on the Adjusted Critical Resource
%SecurityAgents(Numberofdifferentbuildings,Timestep)
%See also RemoveSecurityAgents, AddSecurityAgents
global AreaMatrix
global BuildingMatrix
global CivilianMemoryMatrix
global ResourceInfoMatrix
global FearMatrix
global NumberofSecurityAgents
Size = size(AreaMatrix);
ACR = zeros(1,Numberofdifferentbuildings);
Demand = zeros(1,Numberofdifferentbuildings);
Metabolism = zeros(1,Numberofdifferentbuildings);
PotNumberSecAgents = zeros(1,Numberofdifferentbuildings);
```

```
%going over all nodes
for i = 1:Size(1)
    for j = 1:Size(2)
        for l = 1:Size(3)
            if AreaMatrix(i,j,l,Timestep) == 1
                for k = 1:Numberofdifferentbuildings %Calculating demand for security↙
by every civilian agent
                    a = BuildingMatrix(i,j,k,Timestep);
                    b = CivilianMemoryMatrix(i,j,l,1,Timestep);
                    c = FearMatrix(i,j,Timestep);
                    d = CivilianMemoryMatrix(i,j,l,3,Timestep);
                    e = CivilianMemoryMatrix(i,j,l,2,Timestep);
                    f = ResourceInfoMatrix(i,j,l,2*Numberofdifferentbuildings+1,↙
Timestep);
                    ACR(end,k) = a*b*(1-exp(-(c-d)/(max(e-d,1))*(c-d)/(max(9*f,1))));
                end
                ACR(end+1,:)=0;
            end
        end
    end
end
for k = 1:Numberofdifferentbuildings %calculating number of needed security agents on↙
this timestep per resource
    Demand(k) = sum(ACR(:,k));
    Metabolism(k) = mean(mean(ResourceInfoMatrix(i,j,1,k,1)));
    PotNumberSecAgents(k) = Demand(k)/max(1,Metabolism(k));
end
NumberofSecurityAgents(Timestep,1) = int16(min(PotNumberSecAgents)); %normally it↙
produces a decimal amount of agents
if NumberofSecurityAgents(Timestep,1) > NumberofSecurityAgents(Timestep-1,1)
    plussecagents = NumberofSecurityAgents(Timestep,1) - NumberofSecurityAgents↙
(Timestep-1,1);
    AddSecurityAgents(plussecagents, Timestep);
elseif NumberofSecurityAgents(Timestep,1) < NumberofSecurityAgents(Timestep-1,1)
    minussecagents = NumberofSecurityAgents(Timestep-1,1) - NumberofSecurityAgents↙
(Timestep,1);
    RemoveSecurityAgents(minussecagents, Timestep);
end
end

function SecurityCheck(CivilianTagVariables,Timestep)
%SecurityCheck   Check wether a Security Agent releases a Civilian Agent or arrests a
%Terrorist Agent based on the cultural identity
%SecurityCheck(CivilianTagVariables,Timestep)
global AreaMatrix
global CivilianInfoMatrix
global arrests
Size = size(AreaMatrix);
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep) == 4 %if security agent then we check↙
surroundings
                n=0;
                m=0;
                nr=0;
                [m,n] = CheckSecuritySurroundings(i,j,m,n,nr,Timestep);
                if AreaMatrix(i+m,j+n,1,Timestep) == 2 || AreaMatrix(i+m,j+n,1,↙
Timestep) == 3 %if terrorist agent then something happens
                    culturalidentity = 0;
```

```
                    for l = 1:CivilianTagVariables
                        culturalidentity = culturalidentity+CivilianInfoMatrix(i+m,↙
j+n,1,l,Timestep);
                    end
                    P = abs(((culturalidentity - 22.5)/22.5)^3);
                    x = randi([0,100])/100;
                    if x<P %with unifom probability terrorists is arrested
                        AreaMatrix(i+m,j+n,1,Timestep) = 0;
                        arrests(Timestep,1) = arrests(Timestep,1) + 1;
                    end
                end
            end
        end
    end
end
end

function SecurityWalking(Timestep)
%SecurityWalking   Function to every security agent from their old place to a new↙
place
%SecurityWalking(Timestep)
global AreaMatrix
Size = size(AreaMatrix);
for i =1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep-1) == 4 %Check for Security Agents
                Place = zeros(3,1);
                %Check neighbourhood of Security agent
                for m = -1:1
                    for n = -1:1
                        Dummy = 0;
                        %Check neighbourhood of neighbourhood, to get an
                        %impresseion where the most people are situated
                        if i+m>0 && j+n>0 && j+n<Size(2)+1 && i+m<Size(1)+1 && (m~=0↙
|| n~=0)
                            for mm = -1:1
                                for nn = -1:1
                                    if i+m+mm>0 && j+n+nn>0 && j+n+nn<Size(2)+1 &&↙
i+m+mm<Size(1)+1 && (m~=0 || n~=0)
                                        if AreaMatrix(i+m+mm, j+n+nn, 1, Timestep) ==↙
1 || AreaMatrix(i+m+mm, j+n+nn, 1, Timestep) == 2 || AreaMatrix(i+m+mm, j+n+nn, 1,↙
Timestep) == 3
                                            Dummy = Dummy + 1;
                                        end
                                    end
                                end
                            end
                        end
                        if Dummy > Place(1)
                            Place(1,1) = Dummy;
                            Place(2,1) = i+m;
                            Place(3,1) = j+n;
                        end
                    end
                end
                l=1;
                if Place(2,1)~=0 && Place(3,1)~=0
                    while AreaMatrix(Place(2,1),Place(3,1),l,Timestep) ~=0
                        l=l+1;
```

```
                    end
                    AreaMatrix(Place(2,1),Place(3,1),l,Timestep) = 4;
                else
                    AreaMatrix(i,j,k,Timestep) = 4;
                end
            end
        end
    end
end
end

function SurveillanceData(Numberofdifferentbuildings, Timestep)
%SurveillanceData   Function to calculate and store the surveillane data for active
%terrorists
%SurveillanceData(Numberofdifferentbuildings, Timestep)
global TerroristInfoMatrix
global AreaMatrix
global ResourceInfoMatrix
Size = size(AreaMatrix);
for i = 1:Size(1)
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep)==3  || AreaMatrix(i,j,k,Timestep)==2
                HNW = 0;
                HNWcounter = 0;
                GW = 0;
                %Calculating surveillance data of surrounding nodes
                for m = -1:1
                    for n = -1:1
                        for p = 1:Size(3)
                            if i+m>0 && j+n >0 && i+m <Size(1)+1 && j+n < Size(2)+1
                                GW = GW + ResourceInfoMatrix(i+m,j+n,p,↙
2*Numberofdifferentbuildings+1,Timestep);
                                lookback = 10;
                                if Timestep<11
                                    lookback = Timestep-1;
                                end
                                for PastTimesteps = Timestep-lookback:Timestep
                                    HNW = HNW + ResourceInfoMatrix(i+m,j+n,p,↙
2*Numberofdifferentbuildings+1,PastTimesteps);
                                    HNWcounter = HNWcounter + 1;
                                end
                            end
                        end
                    end
                end
                TerroristInfoMatrix(i,j,k,8,Timestep) = HNW/HNWcounter + GW; %↙
Surveillance data on this node, used to address whether or not to attack
                    lowest = min(TerroristInfoMatrix(i,j,k,1:5,Timestep)); %replacing↙
lowest surveillance data
                if HNW/HNWcounter + GW>lowest
                    dummy = find(TerroristInfoMatrix(i,j,k,1:5,Timestep)==lowest);
                    TerroristInfoMatrix(i,j,k,dummy(1),Timestep) = HNW/HNWcounter +↙
GW;
                    if AreaMatrix(i,j,k,Timestep)==3
                        TerroristInfoMatrix(i,j,k,6,Timestep) = mean↙
(TerroristInfoMatrix(i,j,k,1:5,Timestep)); %mean of biggest 5 surveillance data found
                        TerroristInfoMatrix(i,j,k,7,Timestep) = std↙
(TerroristInfoMatrix(i,j,k,1:5,Timestep)); % std of biggest 5 surveillance data found
                    end
```

```
                end
            end
        end
    end
end
end

function [place] = TerroristOptimalWalk(i,j,Numberofdifferentbuildings, Timestep)
%TerroristOptimalWalk   Function to search for the place the Terrorist
%agent (i,j) wants to go to and returns this as [place]
%[place] = TerroristOptimalWalk(i,j,Numberofdifferentbuildings, Timestep)
global AreaMatrix
global ResourceInfoMatrix
Size = size(AreaMatrix);
z1 = 0;
z2 = 0;
wealth = 0;
for m = -1:1
    for n = -1:1
        if i+m>0 && j+n>0 && j+n<Size(2)+1 && i+m<Size(1)+1 && (m~=0 || n~=0)
            dummy = [];
            for z = 1:Size(3)
                for l = 1:Numberofdifferentbuildings
                    if AreaMatrix(i,j,z,Timestep) == 1
                       dummy(end+1) = ResourceInfoMatrix(i,j,z,↙
l+Numberofdifferentbuildings,Timestep);
                    end
                end
            end
            if max(dummy)>wealth
                z1 = i+m;
                z2 = j+n;
                wealth = max(dummy);
            end
        end
    end
end
place = [z1, z2];
end

function TerroristWalking(amountofcivilians, CivilianTagVariables,↙
Numberofdifferentbuildings, Timestep)
%TerroristWalking   Function to move every Terrorist Agent (active and non-active)↙
from their old place on Timestep-1 to a new place
%on Timestep
%TerroristWalking(amountofcivilians, CivilianTagVariables, Numberofdifferentbuildings,↙
Timestep)
%See also ReplaceTerroristInfo
global AreaMatrix
Size = size(AreaMatrix);
%Going over every node
for i = 1:Size(1)
    for j = 1:Size(2)
        for k=1:amountofcivilians
            if AreaMatrix(i,j,k,Timestep-1) == 2 || AreaMatrix(i,j,k,Timestep-1) == 3
                newplace = TerroristOptimalWalk(i,j,Numberofdifferentbuildings,↙
Timestep-1);
                if newplace(1)>0 && newplace(2)>0
                    ReplaceTerroristInfo(i,j,k,newplace(1),newplace(2),↙
CivilianTagVariables,Numberofdifferentbuildings,Timestep);
                else
                    ReplaceTerroristInfo(i,j,k,i,j,CivilianTagVariables,↙
Numberofdifferentbuildings,Timestep);
                end
            end
        end
    end
end
end
```

C - Functions extended model 1

On the next pages a print of the changes in the code of the first extension of the original model described in section 4.1 can be found. Note that the code should be read as pages from left to right and then from top till bottom. All functions have been saved as separate Matlab code files. The functions shown on the next pages are the only functions that have changed compared to the original code, therefore the other functions that have not been changed and are presented in Appendix B have been used as well.

```
function Attack(Numberofdifferentbuildings,Timestep)
%Attack Function to check whether an active Terrorist Agent does an attack, and performes all necessary consequences
%Attack(Numberofdifferentbuildings,Timestep) checks for the given Timestep whether active terrorist agents are doing an attack and stores information of these attacks
global AreaMatrix
global TerroristInfoMatrix
global ResourceInfoMatrix
global BuildingMatrix
global FearMatrix
global attacks
global AnalysisMatrix
Size = size(AreaMatrix);
for i = 1:Size(1)%going over all nodes
    for j = 1:Size(2)
        for k = 1:Size(3)
            if AreaMatrix(i,j,k,Timestep) == 3 %check for active agents
                mu = TerroristInfoMatrix(i,j,k,6,Timestep);
                sigma = TerroristInfoMatrix(i,j,k,7,Timestep);
                if TerroristInfoMatrix(i,j,k,8,Timestep)>mu + sigma && mu/sigma>0.25 && sum(TerroristInfoMatrix(i,j,k,1:5,Timestep)~=0)==5 %conditions for attack
                    attacks(Timestep) = attacks(Timestep) + 1;
                    fear = 0; %calculation of total fear in area, based on attack
                    xm = 1;
                    alpha = 2.91;
                    U = rand;
                    attacksize = int32(xm/(U^(1/alpha)));
                    for m = -attacksize:attacksize
                        for n = -attacksize:attacksize
                            if i+m>0 && j+n >0 && i+m <Size(1)+1 && j+n < Size(2)+1
                                fear = fear + ResourceInfoMatrix(i+m,j+n,k,2*Numberofdifferentbuildings+1,Timestep);
                            end
                        end
                    end%updating all information in surrounding area
                    for m = -attacksize:attacksize
                        for n = -attacksize:attacksize
                            if i+m>0 && j+n >0 && i+m <Size(1)+1 && j+n < Size(2)+1
                                AnalysisMatrix(1,Timestep) = AnalysisMatrix(1,Timestep) + sum(sum(ResourceInfoMatrix(i+m,j+n,:,:,Timestep)));
                                ResourceInfoMatrix(i+m,j+n,:,:,Timestep)=0; %all resources destroyed
                                AnalysisMatrix(2,Timestep) = AnalysisMatrix(2,Timestep) + sum(BuildingMatrix(i+m,j+n,:,Timestep));
                                BuildingMatrix(i+m,j+n,:,Timestep)=0; %all buildings destroyed
                                FearMatrix(i+m,j+n,Timestep) = fear; %fear created
                                AnalysisMatrix(3,Timestep) = AnalysisMatrix(3,Timestep) + sum(AreaMatrix(i+m,j+n,:,Timestep));
                                AreaMatrix(i+m,j+n,:,Timestep)=0; %all agents destroyed
                            end
                        end
                    end
                end
            end
        end
    end
end
end
```

D - Functions extended model 2

On the next pages a print of the changes in the code of the first extension of the original model described in section 4.2 can be found. Note that the code should be read as pages from left to right and then from top till bottom. All functions have been saved as separate Matlab code files. The functions shown on the next pages are the only functions that have changed compared to the original code, therefore the other functions that have not been changed and are presented in Appendix B have been used as well.

```
function [place] = CivilianOptimalWalk(i,j,k,maxplace1,maxplace2,CivilianTagVariables,↙
Numberofdifferentbuildings, Timestep)
%CivilianOptimalWalk   Function to search for the place the Civilian wants to go to,↙
taking into
%account the place where the maximum encountered resourcevalue is
%[place] = CivilianOptimalWalk(i,j,k,maxplace1,maxplace2,CivilianTagVariables,↙
Numberofdifferentbuildings, Timestep)
%See also AdjustedCriticalResource
global AreaMatrix
global CivilianInfoMatrix
Size = size(AreaMatrix);
z1 = 0;
z2 = 0;
z3 = -1;
vision = CivilianInfoMatrix(i,j,k,CivilianTagVariables+2,Timestep);
search = CivilianInfoMatrix(i,j,k,CivilianTagVariables+1,Timestep); %get the number of↙
the to harvest resource
%Going over all surrounding nodes
for m = -vision:vision
    for n = -vision:vision
        if i+m>0 && j+n>0 && j+n<Size(2)+1 && i+m<Size(1)+1 && (m~=0 || n~=0)  %Check↙
that the Civilian is not going out of bounds
            if abs(i+m-maxplace1)<abs(i-maxplace1)+1 || abs(j+n-maxplace2) < abs(j-↙
maxplace2)+1 %Check that the Civilian is going into the direction of the maximum↙
encountered resourcevalue
                ACR = AdjustedCriticalResource(i+m,j+n,k,search,↙
Numberofdifferentbuildings,Timestep); %Adjusted critical resourcevalue for given node↙
and resource
                if ACR>=z3 %updating newplace
                    z1 = i+m;
                    z2 = j+n;
                    z3 = ACR;
                end
            end
        end
    end
end
place = [z1, z2];
end

function [place] = TerroristOptimalWalk(i,j,k,CivilianTagVariables,↙
Numberofdifferentbuildings, Timestep)
%TerroristOptimalWalk   Function to search for the place the Terrorist
%agent (i,j) wants to go to and returns this as [place]
%[place] = TerroristOptimalWalk(i,j,Numberofdifferentbuildings, Timestep)
global AreaMatrix
global ResourceInfoMatrix
global CivilianInfoMatrix
Size = size(AreaMatrix);
z1 = 0;
z2 = 0;
wealth = 0;
vision = CivilianInfoMatrix(i,j,k,CivilianTagVariables+2,Timestep);
for m = -vision:vision
    for n = -vision:vision
        if i+m>0 && j+n>0 && j+n<Size(2)+1 && i+m<Size(1)+1 && (m~=0 || n~=0)
            dummy = [];
            for z = 1:Size(3)
                for l = 1:Numberofdifferentbuildings
                    if AreaMatrix(i,j,z,Timestep) == 1
                        dummy(end+1) = ResourceInfoMatrix(i,j,z,↙
l+Numberofdifferentbuildings,Timestep);
                    end
                end
            end
            if max(dummy)>wealth
                z1 = i+m;
                z2 = j+n;
                wealth = max(dummy);
            end
        end
    end
end
place = [z1, z2];
end
function TerroristWalking(amountofcivilians, CivilianTagVariables,↙
Numberofdifferentbuildings, Timestep)
%TerroristWalking   Function to move every Terrorist Agent (active and non-active)↙
from their old place on Timestep-1 to a new place
%on Timestep
%TerroristWalking(amountofcivilians, CivilianTagVariables, Numberofdifferentbuildings,↙
Timestep)
%See also ReplaceTerroristInfo
global AreaMatrix
Size = size(AreaMatrix);
%Going over every node
for i = 1:Size(1)
    for j = 1:Size(2)
        for k=1:amountofcivilians
            if AreaMatrix(i,j,k,Timestep-1) == 2 || AreaMatrix(i,j,k,Timestep-1) == 3
                newplace = TerroristOptimalWalk(i,j,k,CivilianTagVariables,↙
Numberofdifferentbuildings, Timestep-1);
                if newplace(1)>0 && newplace(2)>0
                    ReplaceTerroristInfo(i,j,k,newplace(1),newplace(2),↙
CivilianTagVariables,Numberofdifferentbuildings,Timestep);
                else
                    ReplaceTerroristInfo(i,j,k,i,j,CivilianTagVariables,↙
Numberofdifferentbuildings,Timestep);
                end
            end
        end
    end
end
end
```