ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecture with Computer Exercises:
Modelling and Simulating Social Systems with MATLAB

Project Report

Governments, Civilians, and the Evolution of Insurgency: Modeling the Early Dynamics of Insurgencies

Kouzoupis Dimitrios & Rozou Maria & Tsopelas Michail

Zurich
December 2011

Agreement for free-download

We hereby agree to make our source code for this project freely available for download from the web pages of the SOMS chair. Furthermore, we assure that all source code is written by ourselves and is not violating any copyright restrictions.

Kouzoupis Dimitrios Rozou Maria Tsopelas Michail

I

Contents

1 Abstract

The aim of this is project is to create an accurate model for simulating the early dynamics of insurgency against the established government. We constructed an agent-based simulation, in which the agents are the simple people (civilians),the armed supporters of the government (soldiers) and the leaders of the insurgency (leaders). The main idea of our simulation is that when a rebellion occurs, insurgents attack the government forces and consequently the latter will respond to their attack with counter attacks. Furthermore, the success of these counter attacks depends both on the soldiers' effectiveness to capture insurgents and also on their accuracy to avoid collateral damage. So, depending on these two parameters, the counter attacks may lead either to the death or capture of the insurgents, but they could also augment the anger of the nearby civilians because of the collateral damage that they caused. In this case, civilians become angry and they may become new insurgents. Thus, the government must find an efficient combination of soldiers' accuracy and effectiveness in order to succeed in its goal, namely to suppress the rebels as fast as possible. In our project, we attempted to investigate the relations between several factors/parameters that could potentially affect the dynamics of such insurgencies. For this purpose, the model proposed by Bennett ('Governments, Civilians, and the Evolution of Insurgency: Modelling the Early Dynamics of Insurgencies ', *Journal of Artificial Societies and Social Simulation,* Volume 11, No. 4 7), is further elaborated and enhanced with new parameters, leading to more accurate representation of a real world scenario.

2 Individual contributions

All team members contributed equally to this project.

3 Introduction and Motivations

3.1 Introduction

Since the beginning of civilization, people had the tendency to organize themselves into groups in order to fulfil their own goals and their own ambitions. Many times, these goals were opposed to government's actions and this made them angry enough to rebel against it so as to assert their rights. These groups of people, had as main objective to impose their own agenda, believing that all their problems were the consequences of some incorrect government decisions. Such revolutionary acts are by far more frequent and intense nowadays. Global recession, social injustice, the financial gap between wealthy and poor, corruption and low quality of life are some of the reasons that have forced people to rebel against governments.
Trying to understand and model the dynamics of such insurgencies is of course extremely hard. Most of the times, especially nowadays, there are hundreds of parameters and forces that interact during a rebellion and can influence its outcome. Many recent examples can be cited. During the 20th century for instance, the decolonisation resulted in many bloody conflicts, where the colonies of many superpowers, attempted to gain their independence (India, Philippines, Algeria etc.). More recently there are many civil conflicts, were people are fighting for their civil and political rights (eg. Libya, Syria, Egypt) and attempt to overthrow corrupted dictators.
We could in general divide the factors influencing the outcome of an insurgency into two main categories:

1. **Exogenous factors**: these relate to external forces which can affect the insurgency one way or another. The international community is one of these major factors, but there are others as well, like foreign alliances, influences, financial condition etc.

2. **Internal factors**: these relate to forces inside the country that the revolution takes place, such as the political, economical and legislative situation, the state of the armed forces, the geography of the area, the population etc.

In our subsequent analysis, we focus only on the 2nd category, since it would be too complicated to attempt to simulate external forces.
The main characteristic of insurgencies, is that majority of them actually fail to achieve their main goal because they are simply defeated. This seems pretty reasonable since especially at their early stages, insurgencies are extremely vulnerable: they deal with a much more organized and better equipped enemy, with far better infrastructure and firepower.
Historians and military tacticians, have long debated on the best way to handle

insurgencies. There are two main doctrines: the **big stick** or **direct-approach** doctrine,focuses on defeating the insurgents on the battlefield. This is widely considered as the conventional approach. The second **indirect approach**, recognizes that it is essential also to attack the support of the people for the insurgents, in an attempt to alienate the insurgency from the majority of the population. One of the most famous examples of direct approach was the Vietnam War, which ended with the humiliation of the US. Since then, there has been a shift towards a more indirect approach, which focuses on minimizing civilian loses and generally restrain the use of force as much as possible.

3.2 Motivation

In his work, Bennett focused mainly on understanding what makes or breaks an early insurgency. He supported that there are two critical parameters: how effective are the government forces in eliminating active insurgents, and how accurate they are when doing so (by avoiding collateral damage). Based on what we previously mentioned regarding the different approaches, his work attempts to explore the consequences of the direct and indirect approach. What he discovered, is that in the long run, it is more important to adapt a more indirect approach, than using brute force to fight an insurgency. In fact, when using excessive force, governments alienate themselves from the population, pushing more and more people into the insurgency; effectively helping the insurgents. Accuracy is much more important than effectiveness, because high accuracy results into fewer civilian casualties and anger; thus even if it takes more time to defeat the insurgents (meaning low effectiveness), the insurgency will eventually be defeated, since it will never become self-sustaining. Even though his analysis is thorough, we felt that he focused on very few parameters so as to accurately simulate the dynamics of an insurgency. Moreover, we believe that in the relevant model, the odds were far against the insurgents, which nowadays might not be particularly true. What we tried to do, was to give the insurgents some more capabilities, making it a bit harder for the government to actually defeat them and bringing the simulation closer to reality. This was done by making the model a bit more complex, namely by inserting new parameters that could affect the outcome of insurgencies. We wanted to make the model more customizable with main objective to provide a more realistic model for the dynamics of insurgencies, that could be also tested on real data. To sum up, our goals were the following:

1. Attempt to elaborate on why accuracy is more important than effectiveness , when dealing with long-term insurgencies. *Why and when is this true?*

2. Attempt to simulate many factors that influence insurgencies, such as the media, the weapon technology of government forces, the characteristics of the

population etc.

3. Draw conclusions on the dynamics of insurgencies.

4 Description of the Model

4.1 Bennett's Model - The starting point

Based on the work of Bennett, initially our model contained two types of agents: *soldiers* and *civilians*.Soldiers represent the government forces which are responsible for responding to the attacks of the insurgents. The main characteristics of a soldier is his *effectiveness* in capturing or killing an insurgent and his *accuracy* when attempting to kill or capture the rebel (i.e avoiding collateral damage).
Civilians represent the ordinary people of a country and they also have some basic characteristics. These are their *fear* and *anger* against the government. Also, each civilian is characterized by a violence *threshold*, which represents his tendency to react violently against the government. Civilians with higher violence thresholds will be more reluctant to take up arms against the government, whereas those with low violence thresholds will turn into insurgents with little provocation.
There are two driving forces behind the model:

1. The first force, is that a civilian becomes a potential insurgent if his anger level surpasses both his fear and threshold levels. In this case, this civilian who is now insurgent may attack any nearby soldier. In order to separate the insurgents between those who have not made an attack so far and those who have, we have two categories of insurgents: the *latent* and the *active* insurgents respectively. Any latent insurgent, with soldiers in his neighbourhood, will attack a soldier and thus, he will become an active insurgent.

2. The second force, is the soldiers' response to the insurgents attacks. After a soldier has been attacked, he can respond (with a probability equal to his *sensitivity*) and attempt to kill or capture the insurgent who attacked him. The counter attack can have two consequences: firstly, it can remove the attacking insurgent from the world (with a probability equal to the *effectiveness*) of the counter attacking soldier. Secondly, regardless of whether he managed to kill or capture the insurgent, a counter attack can inflict collateral damage (by injuring neighbouring civilians). This depends on the soldiers *accuracy*. Based on the amount of collateral damage, the anger levels of neighbouring civilians can increase, resulting perhaps in more civilians rebelling against the government.

All the above can be parametrized in a simulation model. As we already said, each soldier has a degree of *effectiveness* and *accuracy*, which capture his ability to kill or

capture insurgents and avoid collateral damage, respectively. Furthermore, the fact that collateral damage increases the fear and anger levels of civilians, is modelled through a rate of anger and fear increase.
Figure 1 shows the main steps of the simulation as proposed in the work of Bennett. The main points are the following: first, an insurgent is randomly chosen to attack a soldier. If such an insurgent can be found, then the attack is performed and the attacked soldier counter attacks. With probability equal to the effectiveness, the insurgent who attacked, is removed from the simulation. Then, with probability equal to 1-accuracy, each civilian in the neighbourhood of the insurgent, is injured. Then, the levels of anger and fear of the injured civilians are updated accordingly (proportionally to the number of civilians injured). Notice, that both anger and fear levels increase at the same time. The simulation continues until there are no more insurgents or there is no soldier in range of an insurgent, or a maximum number of iterations has been reached (this usually means that the insurgency is successful).
Another important thing to note, is that soldiers cannot die during the attack of an insurgent. This is done to simulate the different capabilities of the two adversaries. Indeed, governments have a much bigger manpower pool and thus individual loses of soldiers can be easily filled with new recruits. Hence why the only possible ways for a simulation to end, is either by killing all the insurgents, or by reaching a maximum number of iterations.

4.2 The proposed extended model:

Our work is of course based on the model we just described in Figure 1. However, we felt that this model, although it captured the basic points, it missed quite a few important parameters and also made some very simplistic assumptions.

1. It does not take into account the diversity of the population, such as different ethnic, political and religious backgrounds.

2. When it comes to rebelling against the government, the wealth of each individual is very important. Indeed, we can scarcely find examples of rich people rebelling against governments; it is the poorer classes that are more inclined in doing so.

3. It does not consider the importance of media, and information diffusion. In terms of parameters, having the same grid size for all kinds of agents interactions is unrealistic. One should be able to model for example, how many civilians are affected by a soldier's counter attack and how the news of civilian injuries are spread to the population.

4. It assumes that all soldiers have the same rates of effectiveness and accuracy.

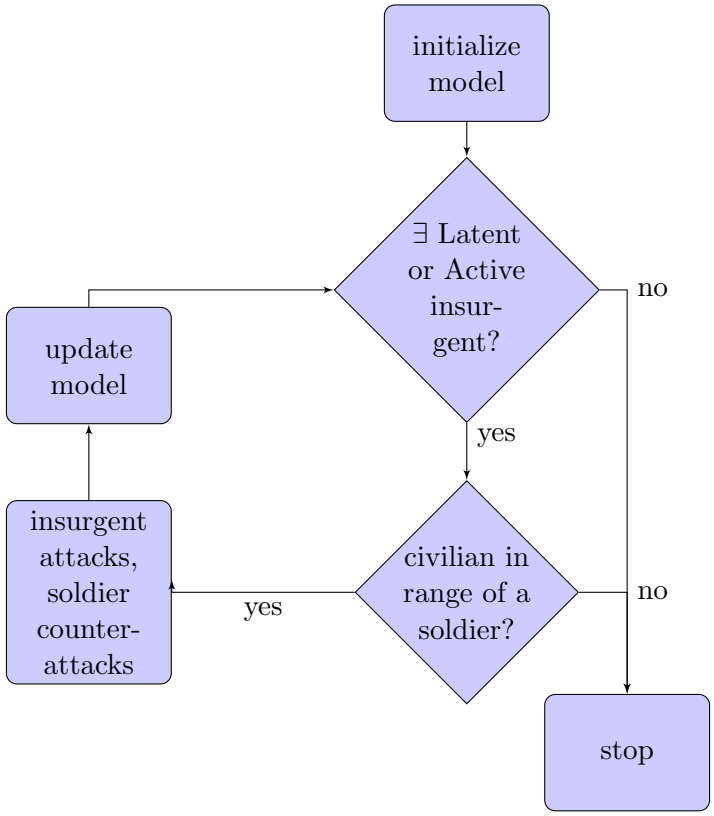

Figure 1: Simplified flow chart of the simulation overview.

5. Only civilians who are injured actually change their fear and anger levels.
6. All civilians are affected in the same way when it comes to changing anger and fear levels.
7. There is no cooperation between insurgents.
8. Only two actions for interaction: insurgents attack and soldiers counter attack.
9. Lastly, civilians cannot move around the map, resulting in many insurgencies ending because no interaction between agents is possible.

From the above points, one can see that there is plenty of room for expanding the model, trying to make it behave more realistically.

Interaction: To more realistically model the interaction between the different agents, we decided to adopt three different interaction grids.

1. The *Attack grid* defines the area in which an insurgent can attack a soldier.
2. The *Collateral grid* defines the area in which civilians may be injured by a soldier's counter attack.
3. The *Influence grid* defines the area around an insurgent, in which the anger and fear levels of civilians in that area are changed according to the civilians injured.

The reason why we introduced these different grids was to make our simulations more realistic and customizable. Our main concern was that not only the injured civilians should change their attitude towards the government but also the nearby civilians who were witnesses of the event or heard the incident on the news. The bigger the Influence grid the more we assume that the media inform citizens about the collateral damage caused by government actions.

Agents To make the agents a bit more realistic, and introduce some variance in our model, we made the following changes:

1. **Soldiers**, now do not share the exact same values for effectiveness and accuracy. Instead these values are drawn from a $N(\mu, \sigma)$ distribution, where usually σ is some small number(0.01). This, although not game breaking, brings the model closer to reality.
2. **Civilians** have been overhauled. We introduced three new classes of civilians, namely *Wealthy*, *Middle* and *Poor*. The three classes draw their initial anger, fear and threshold values from different normal distributions. This allows us to further customize and parametrize our model, and to more accurately reflect reality. This seems natural: poor civilians should have a lower violence threshold than rich civilians. Also they should be angrier and less afraid of the government. The percentage of each class contribution to the total population is of course configurable (meaning that we can build a simulation with more poor or rich civilians).
   Furthermore, we gave insurgents the ability to move around the map. This was done, mainly to prevent insurgencies from ending with neither side winning, since usually, after a couple hundred simulation steps, insurgents could not find any soldiers in their *Attack Grid* to attack. Thus, when no insurgent is in range to attack a soldier, we randomly choose one, and move him near a soldier. This keeps the simulation going. Of course, this ability adds many new interesting ideas that can be implemented (like for example agents cooperation for avoiding soldiers, etc.).

3. A new type of agent was introduced, in order to make things a bit more complicated: **leaders**. At the beginning of each simulation, a small (typically between 5 to 10) number of leaders is inserted into the world. Leaders have several important characteristics: first of all, they can neither attack nor being attacked. Leaders are the figures that inspire and represent the insurgency. They can only perform two actions: either they can **recruit** a candidate civilian, or they can spread **propaganda**, increasing the support of local civilians towards the insurgency. Each leader has different characteristics and abilities: one might be very adept at recruiting new insurgents, whereas another might be very good at gathering support for the insurgency. This is implemented by drawing values from normal distributions with high variance. Leaders have values which determine the probability of successfully recruiting a civilian, and/or perform a propaganda action as well as the degree at which the civilians are affected by such an action.

Updating the agent's state: An important change concerns the way each civilian's state is updated, after a soldier counter attacks. Bennetts' model made the assumption that only injured soldier change their fear and anger levels, which did not seem natural at all. So, what we did, was to change the anger and fear levels of every civilian in the *Influence Grid* of an attacking insurgent. When updating their state, we distinguish between civilians who were injured during the counter attack, and civilians who were not injured. Our main approach was that civilians who are injured tend to become more afraid than angry, whereas neighbouring civilians who learn of the news, tend to become more angry than fearful. Furthermore, the update of the fear and anger levels is further influenced by the class in which the civilian belongs to. Again here, we propose that rich civilians increase their fear levels faster than their anger levels, whereas poor civilians do exactly the opposite; their fear levels are less affected, whilst their anger increases faster than that of any other class.

4.3 Simulation Overview

In our simulation, there are three types of agents:

- **Soldiers.** Soldiers represent the government forces and they can be attacked by the insurgents. Their main characteristics are their effectiveness to kill or capture an insurgent, their accuracy to avoid injuring nearby civilians and their sensitivity, which is how probable is the fact that they will respond to an attack.
- **Civilians.** Civilians represent the ordinary people and each of them has some

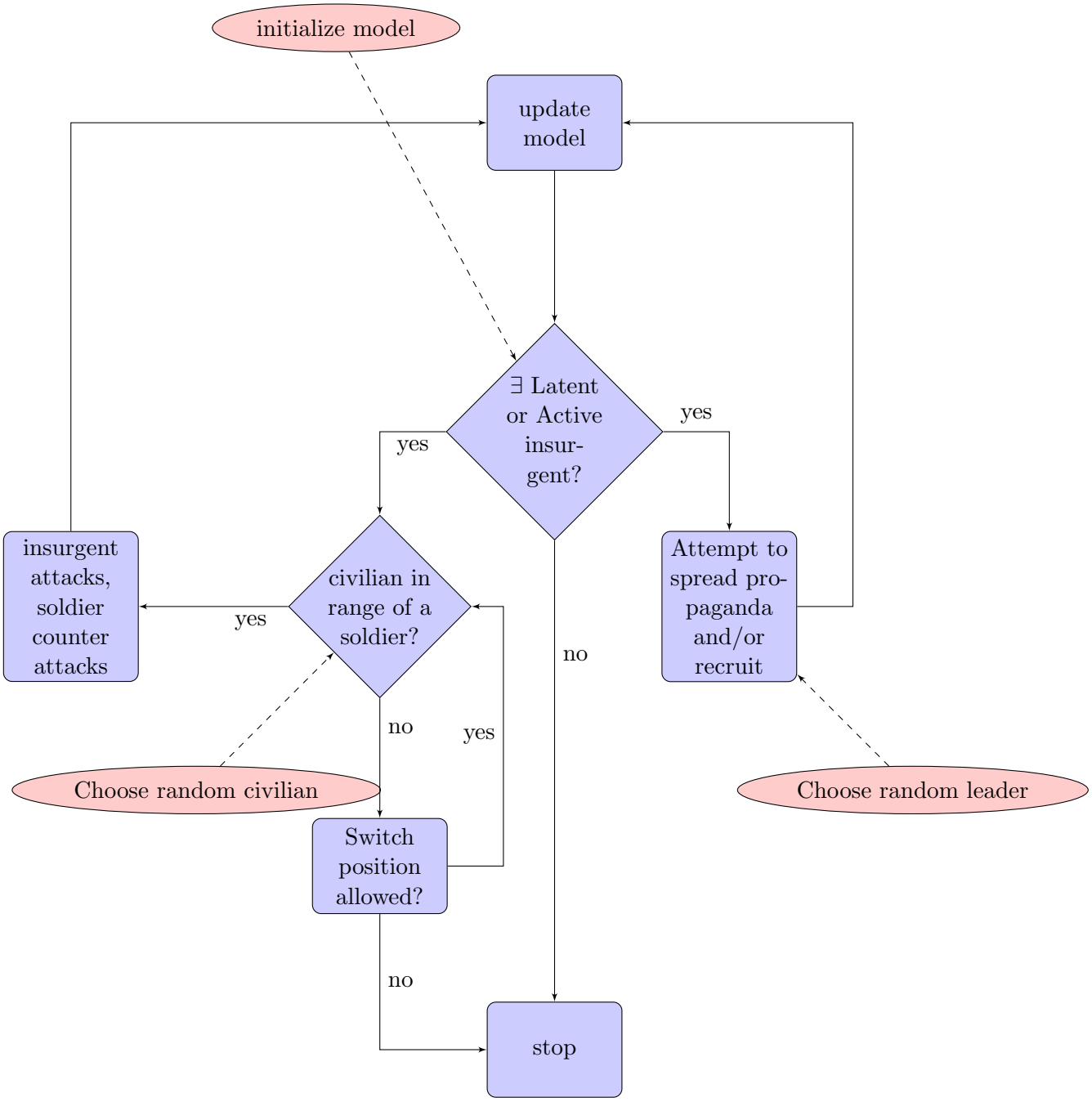

Figure 2: Flow chart of the implemented simulation model.

specific characteristics, like a degree of anger at the government, a degree of fear of the government, a violence threshold and their level of wealthness.

- **Leaders.** Leaders are responsible for recruiting a civilian and for spreading propaganda among civilians, in order to increase the support of civilians towards the insurgency.

An important group of people between these three agents is the insurgents. Those are a specific group of civilians who have become more angry than afraid and also their anger surpasses their violence threshold. They are the only one who can attack to the government forces. We seperated them in two different categories: the *latent insurgents*, who are civilians that are willing to participate in an insurgency and the *active insurgents*, representing those who have already make at least one attack. So, if a latent insurgent makes an attack, he becomes an *active insurgent*.

The main idea of our simulation is that in every time step, we choose randomly one active or latent insurgent to attack to a soldier who is within his *Attack Grid*. If there is not any soldier within this range, and if the insurgent is able to change position, he searches for a soldier to attack. Then, the soldier that received an attack is going to respond with another attack. This counterattack aims either to the death of the insurgent or just to capture him. In order to be close to real world conditions, it is not at all sure that this soldier will succeed in his goal. Not only may he be incapable of capturing this insurgent but he may also cause collateral damage during his way to kill the insurgent, which is the number of civilians that get injured during the counterattack of the soldiers. That is why we have defined these two characteristics of every soldier, the *effectiveness* and the *accuracy*. Thus, the insurgent who is attacked by the soldier will be killed or badly injured with probability equal to this effectiveness and he will then be removed from the world.

Also, civilians who are within the *Collateral Grid* will get injured with probability *1 - accuracy*. Civilians who are within the *Influence Grid* saw all these violent actions will become more afraid of the government for safety reasons and they will get angry analogously to the degree of the provoked damage, i.e analogously to how many civilians are injured during this counterattack. Furthermore, the update of the fear and anger levels depends on the class that they belong to. For example, rich civilians increase their fear levels faster than their anger levels but poor civilians increase their anger faster than any other class.

In addition to these, in every time step we choose randomly one leader. He can neither be attacked nor attack to a soldier. He is responsible for recruiting civilians who are more angry than afraid, and also for spreading propaganda between civilians. A propaganda action increase the anger levels of the nearby civilians and decrease their fear. These two characteristics of leaders, make their role very important for

the success of the insurgency in every time step, because firstly recruited civilians become latent insurgents and secondly because the insurgency gains more support. In, the following figure we can see a typical example of agent interactions.

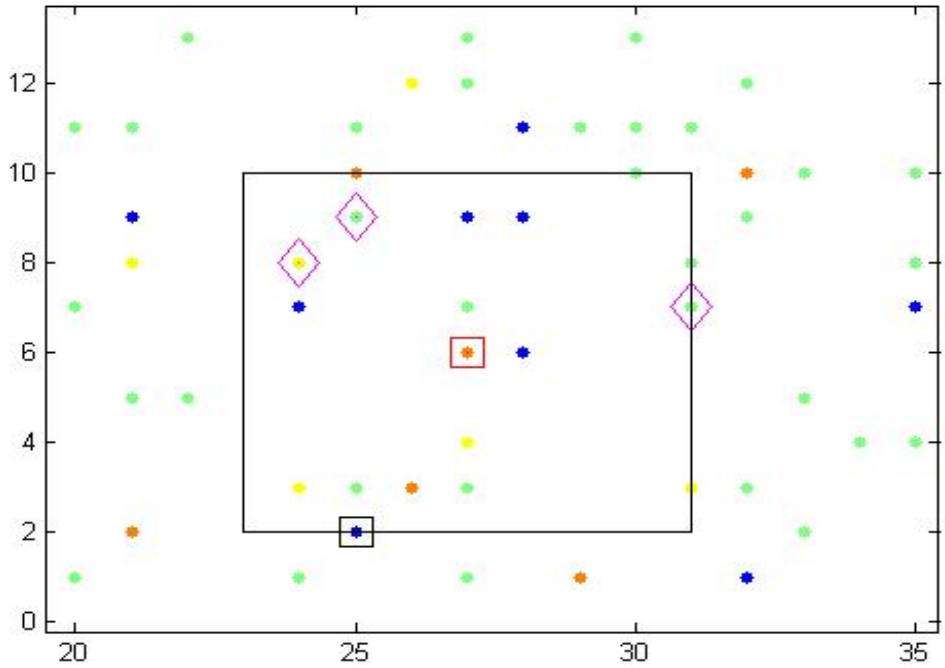

Figure 3: Typical example of agent interactions. The orange rectangle depicts the insurgent performing an attack on the soldier with the small blue rectangle. The magenta diamonds, illustrate the civilians that got injured by the soldier's counter attack. The *Collateral Grid* is depicted with the large black rectangle.

Furthermore, in our simulation, it was impossible for a soldier to die when he was attacked by an insurgent. We made the assumption that even this soldier died, the government replaced him immediately and the simualtion continues with the same number of soldiers. This happened because of the government's ability to recruit new soldiers immediately at the early stage of an insurgency. On the other hand, civilians could die when a soldier counterattacks to them. For the replacement or not of this civilian, we implemented the variable *afterlife* and if we believe that there is not any life after death, then we did not replace this civilian, but if we believe in reincarnation, we generate a new civilian with new characteristics somewhere else in our world frame. Last but not least, the termination condition was achieved if

there is not any latent or active insurgent in the world frame, or if there was not any soldier in the *Attack Grid* of the insurgent and he could not change position. Lastly, the software used for the simulation was the MATLAB Programming Language.

5 Implementation

5.1 Agents

The final model of our world contains as we have mentioned three main types of agents. Soldiers, civilians and leaders. In this section we will present in more detail how these agents are implemented.

Starting from the soldiers, we have adopted Bennetts' point of view and we have chosen a fixed number of 100 soldiers in all our simulations. This number is steady throughout the simulation since insurgents are incapable of killing a soldier or equivalently, we can assume that dead soldiers are immediately replaced by the government. This assumption is perfectly valid, especially in the first stages of a revolution which is exactly what we want to simulate. Each soldier is uniquely defined by a structure containing the characteristics showed in Table 1.

Table 1: Soldier structure

Parameters	**Structure field**
Position in map	S.x & S.y
Level of accuracy	S.accuracy
Level of effectiveness	S.effectiveness
Sensitivity	S.Sensitivity
Colour	S.colour

The levels of accuracy and effectiveness are drawn from a normal distribution with variance equal to 0.05, instead of being identical for all soldiers. Our purpose is to make the simulation a little bit more realistic by implying that even if all soldiers are trained the same way, we can not possibly expect that they have exactly the same characteristics. On the other hand, the sensitivity value is always set to 1, which means that soldiers always respond to insurgent attacks. Finally we have assigned the blue colour to the corresponding parameter to represent the soldiers when we display a snapshot of the world.

Perhaps the most significant type of agent in our simulation is the civilian. There are typically 500 civilians in our simulations setting a reasonable ratio in our world. Each one of them is again represented by a structure with four main individual

characteristics: anger, fear, violence threshold and wealth. The first three variables range from 0 to 1 and since they are samples of statistical distributions, a threshold is used to ensure that all results fall into this range. The wealth variable indicates whether civilians are *poor*, *rich* or belong to the *middle class*. Its value drastically affects their behaviour towards the government since the financial condition of each person influences the mean value and variance of all three other variables (fear, anger, violence threshold) in a reasonable way. The default values are analytically described in Table 2. The fields of the civilian structure, is presented in Table 3.

Table 2: Mean and Standard Deviation of Civilians' parameters

	Mean			Standard Deviation		
	Rich	Middle	Poor	Rich	Middle	Poor
Anger	0.15	0.25	0.45	0.125^2	0.125^2	0.125^2
Fear	0.7	0.5	0.25	0.25^2	0.25^2	0.25^2
Violence Threshold	0.8	0.5	0.4	0.25^2	0.25^2	0.25^2

Table 3: Civilian structure

Parameters	**Structure field**
Position in map	C.x & C.y
Level of anger	C.anger
Level of fear	C.fear
Violence threshold	C.thres
Wealth	C.wealth
Colour	C.Colour

In our general approach, in order to separate civilians into social classes, we use a uniform distribution and we perform a classification based on the interval of the result. If the value is between 0 and 0.2, the civilian is characterized as *Poor*. In other words, we assume that 20% of the population is living in poverty. If the result is between 0.85 and 1 civilians are considered to be *rich* and finally, in all other cases they belong to the middle class. This would also enable researchers to use country specific data to simulate hypothetical scenarios.

The final type of agent in our simulations is the *leader*. Leaders are special agents that do not directly attack soldiers; instead they have a key role in the support of an insurgency, which involves recruiting new members and helping spread the revolutionary ideas among civilians. A detailed description of their actions can be found in the next section.The fields of the structure describing a leader is shown in Table 4. Note that each leader is distinct, with different characteristics. In particular,

this means that the fields *L.influenceChance, L.recruitChance, L.influenceRange, L.influenceRate* are drawn from normal distributions. The typical values of these parameters are shown in Table 5 :

Table 4: Leader structure

Parameters	**Structure field**
Position in map	L.x & L.y
Probability to perform propaganda actions	L.influenceChance
Probability to recruit a civilian	L.recruitChance
Range of civilians affected by the propaganda	L.influenceRange
Impact of propaganda to civilians	L.influenceRate
Colour	L.Colour

Table 5: Leader Parameters

Parameter	**Mean**	**Standard Deviation**
L.influenceChance	0.1	0.001
L.recruitChance	0.1	0.001
L.influenceRange	20	2
L.influenceRate	0.08	0.001

Display: Agents appear in our plots with different colours in order to distinguish all their different types. Soldiers are represented by *blue* dots while the colours of civilians vary depending on their state. If they are latent insurgents, meaning that the are ready to attack but they have not performed any actions yet, they are *orange*. If they have become active insurgents they turn red and if their anger is higher than their fear but their violence threshold is not surpassed, they are *yellow*. Yellow civilians are not insurgents yet but they still are angry at the government. In any other case, civilians are not consider to be a threat and they are shown in *green* colour. Finally, we use magenta colour to depict the leaders in our map. All these colour correspondences are grouped in Table 6.

5.2 Agent Actions-Effects

In the previous section, we explained the structure of the agents in our world. Now we will proceed with describing what exactly each agents does.

Soldiers Soldiers are the supporters of the government. Their goal is of course to neutralize all the insurgents. Soldiers can only perform one action, namely to

Table 6: Colour Correspondences

State	**Colour**
Soldiers	Blue
Active Insurgents	Red
Latent Insurgents	Orange
Anger > Fear & Anger < Violence Threshold	Yellow
Anger < Fear & Anger < Violence Threshold	Green
Leaders	Magenta

counter attack against the attacking insurgent. Counter attacking means that after an insurgent has exposed himself (by attacking a soldier), then this soldier has the chance to eliminate the insurgent (in practice, that would mean either killing or capturing him). In the general case, a soldier will not always respond to a insurgents' attack. In fact:

$$P(\text{soldier responding to insurgents' attack}) = S.sensitivity$$

There are two types of effects, when a soldier counter attacks:

1. With a probability equal to $S.effectiveness$, he will kill or capture the insurgent, effectively removing him from the simulation.

2. A counter attack may injure nearby civilians with a probability that depends on the parameter $S.accuracy$. Namely,

$$P(\text{soldier injures a nearby civilian}) = 1 - S.accuracy$$

The impact of a counter attack is the following. Each and every civilian that exists in the range of the soldier's collateral grid has a probability to be injured that is equal to $1 - accuracy$ as we have just mentioned. The total number of injured civilians is stored into a variable called *damage* because it directly affects the fear and anger increment of all civilians within the Influence grid. Moreover, we assume that the impact would be more intense on injured civilians than on people that were just informed about the incident and also that different social classes have different reactions and are affected in different ways. To implement this idea we introduced a number of parameters called modifiers which are presented in Table 7. According to these modifiers, the state of every civilian in the Influence grid is updated as follows:

$if \quad injured == 1 \quad then$

$fear = fear + fearModifier * fearBaseModifierVictim * (1 - fear)$
$anger = anger + angerModifier * angerBaseModifierVictim * damage * (1 - anger)$

$else$

$fear = fear + fearModifier * fearBaseModifierSpectator * (1 - fear)$
$anger = anger + angerModifier * angerBaseModifierSpectator * damage * (1 - anger)$

Table 7: Base modifiers

Modifier	Default value
Fear Base Modifier (Victim)	0.1
Fear Base Modifier (Spectator)	0.05
Anger Base Modifier (Victim)	0.05
Anger Base Modifier (Spectator)	0.05
Fear Modifier (Rich)	1.1
Fear Modifier (Middle)	1
Fear Modifier (Poor)	0.9
Anger Modifier (Rich)	0.8
Anger Modifier (Middle)	1
Anger Modifier (Poor)	1.2

Insurgents As we said, civilians become insurgents when:

$$C.anger \geq C.threshold$$
$$C.anger \geq C.fear$$

Insurgents can mainly perform two actions:

1. Firstly, they can *attack* a nearby soldier. The insurgents only look for soldiers inside the area specified by the *Attack Grid*. This attack might trigger a *counter attack* by the defending soldier.

2. Secondly, insurgents can move around the map. This will only happen when there is no available insurgent that can *attack* a nearby soldier. This action was implemented mainly to avoid simulations ending because there is simply no nearby soldier for an insurgent to attack. However, we believe it can be used for many different purposes, for example, to allow insurgents to escape or cooperate.

Leaders As we said, *leaders* focus on the non military side of the conflict. They are there to help the insurgents gather support, and for this, we implemented two main actions for them:

1. The first, is the ability of a leader to *recruit* a civilian. Recruitment is only applicable for eligible civilians, that is civilians who are more angry than feared (but not yet latent or active insurgents, obviously). The probability that a leader will successfully recruit such a civilian is determined by L.recruitChance. A recruited civilian, immediately becomes a *latent* insurgent.

2. The second, is the ability of a leader to spread *propaganda* throughout the population. The probability that a leader will successfully carry out a propaganda is determined by L.influenceChance. Given that a leader performs a propaganda action, then all the civilians in the area determined by L.influenceRange, have their *anger* and *fear* levels changed according to the following equations:

$$C.anger \leftarrow C.anger + L.influenceRate * (1 - C.anger)$$
$$C.fear \leftarrow C.fear - L.influenceRate * (1 - C.fear)$$

   In other words, a propaganda action will increase the anger levels of the nearby citizens, and decrease their fear. Note that each leader has his traits randomly generated

5.3 Simulation Details

1. *Soldiers:* There are 100 soldiers in our simulation. Each soldier is defined by two parameters that depict his position in the simulation space. He is declared as a structure with three separate characteristics: effectiveness, accuracy and sensitivity. In order to be as close to the real world conditions as possible, we did not assume that every soldier has the same characteristics, but on the contrary, the effectiveness and accuracy of every soldier are drawn from a normal distribution with standard deviation equal to 0.05. Since we draw those parameters from normal distributions and we want them to have values in the range of $[0, 1]$, we of course truncate all the values that fall outside the mentioned range. An close to 1 means that the soldier has a large probability of killing an insurgent during his counter attack. Similarly, the closest the accuracy to 1 is, the less collateral damage will be caused. In other words, each civilian in the collateral grid, has a probability of 1-accuracy of being injured during the soldiers' counter attack. Sensitivity was considered a constant equal to 1 in order to ensure that a soldier will respond to the attack.

2. *Civilians:* There are 500 civilians in our simulation. Each civilian is a structure with four different characteristics: anger, fear, threshold and wealth. Our initial conditions for these variables were taken from normal distributions with mean and standard deviation as shown in Table 2. The fourth parameter shows the social class of each civilian. We assumed that 20% of our population is living in poverty so they belong to the *Poor* class, 15% of our population is rich, so they belong to the *Rich* class and all the other people belong to the *Middle* class. As in the case of soldiers, we used a threshold in our values in order to be within the acceptable range.

3. *Leaders:* There are usually 5 leaders in our simulation. As mentioned before, leaders cannot attack to the soldiers neither be attacked by them. They are responsible for recruiting a candidate civilian and for spreading propaganda among civilians. Their main goal is to increase civilians' support towards the insurgency.

The main algorithm of our simulation is the following (see Figure 2):

1. Model initialization. Map is initialized, soldiers, civilians and leaders are randomly distributed. Each agent is assigned with his initial values, concerning fear, accuracy, sensitivity, anger, violence threshold etc. Civilians with appropriate levels of fear and anger are updated and displayed as latent insurgents.

2. while (flag==true)

   (a) If leaders are allowed, then we randomly pick a leader. Then, this leader has a probability equal to $influenceChance$ of successfully carrying out a propaganda action, in which case all the civilians in his $influenceRange$ have their anger increased and their fear decreased, depending on the leader's $influenceRate$. Furthermore, with probability equal to $recruitChance$, the leader performs a recruit action; a random civilian whose anger is greater than his fear but less that his violence threshold, is turned into a latent insurgent (by setting his anger level, a bit higher than his violence threshold).

   (b) A latent or active insurgent is randomly chosen to attack a randomly selected nearby soldier.

   (c) The attacked soldier, counter attacks.

   - The insurgent is neutralized with probability equal to the soldier's effectiveness. We can choose either to replace or not the insurgent, depending on how we initialize our model. The newly introduced civilian, has random and uncorrelated (with the removed civilian) values for his parameters and is positioned in a random cell of the grid.

- Each civilian in the collateral grid surrounding the insurgent is injured with a probability equal to 1-accuracy.
- Each civilian in the influence grid surrounding the insurgent has his fear and anger values updated accordingly. Injured civilians will have different update modifiers than the others. Moreover, modifiers depend on each civilian's social class.
- Map is updated to reflect the new state (remove attacking insurgent, add a new civilian, change the display colours of civilians- if needed).

(d) Check for terminal conditions: if no insurgent is alive, or the maximum number of iterations has been reached, then flag=false. Also, if no civilian is in range of a soldier, flag=false, unless we allow civilians to move around the map, in which case we randomly select a civilian and place him close to a soldier.

Some of the basic parameters of our model are shown in Table 8.

Table 8: Basic Model Parameters

Parameter	**Definition**	**Default Value**
Map size	Dimensions of map grid	50
Attack Grid	Size of neighbourhood for insurgents looking to attack	9
Collateral Grid	Range for collateral damage to civilians	9
Influence Grid	Range for the spread of news after a counter attack	15
Accuracy	1-accuracy = prob. of injuring any civilian withing Collateral Grid	0.8
Effectiveness	prob. to kill an insurgent during a counterattack	0.8
Sensitivity	prob. of a soldier reacting to an insurgent's attack	1
Initial Anger	Initial anger of civilian toward the government	$N(\mu, \sigma)$
Initial fear	Initial fear of civilian	$N(\mu, \sigma)$

6 Simulation Results and Discussion

6.1 Experiments - Simulation Results

In this part, we will present and discuss the key observations derived from the simulations. We will split this section into four parts:

1. Compare our results with the results presented in the work of Bennett(2008).
2. Present further results from our extended model, such as the inclusion of different interaction grids, replacement of civilians, changing of position etc.

3. Discuss the results.

Our main focus in such simulations are two things: firstly, how long do the insurgencies last on average and secondly, how many deaths occur on an "average" insurgency. For all the subsequent experiments, we used the parameter values as shown in Table 9(unless otherwise specified).

Table 9: Parameter values for experiments

Parameter	**Value**
Grid Size	50
Number of Civilians	500
Number of Soldiers	100
Attack Grid	9
Collateral Grid	9
Influence Grid	15
Sensitivity	1
Fear Increment	0.1
Anger Increment	0.05

6.1.1 Repeated Simulations - No civilian replacement

Our first experiment focused on running a batch of simulations for all possible combinations of effectiveness and accuracy levels, $accuracy \in [0, 1]$, $effectiveness \in [0.1, 1]$. For each combination, we run a total of 10 simulations, and afterwards we averaged the results. Our model was kept as simple as possible at first: we did not have any insurgent leaders, nor civilian replacement and we did not allow change of positions for the insurgents. Notice that there are two ways for a simulation to end: either the insurgency is defeated (namely, no more latent insurgents remain), or a maximum number of iterations is reached (in our case, 5000 iterations). Results are shown in Figures 4, 5. Notice that out of the 1100 simulations, only 15 where actually defeated, the rest 1085 were left undecided since usually the insurgents end up being out of the range of soldiers and thus cannot attack them.

Next, we do the same experiments again, this time allowing the insurgents the freedom to move around the map when they are out of range of all soldiers. This way, the insurgency's outcome cannot be *undecided*, unless it is not defeated before reaching the maximum number of iterations. Such a simulation setup seems of course much more closer to reality, than the previous one. Results are shown in Figures 6, 7 and in tabular form in Table 10. Notice that in this case, out of the 1100 simulations, only 121 were undecided, whereas the rest 979 were defeated.

Table 10: Results: No civilian replacement

	Average Time Steps	**Average Deaths**
Civilians can move	1808	354
Civilians cannot move	1335	337

We now move on to even more realistic experiments.

6.1.2 Repeated Simulations - Civilian Replacement

On the previous simulations, when a civilian (an insurgent) died, he was removed irrevocably from the simulation, and the total population decreased by one. However, this is somewhat counter-intuitive, since it means that governments could defeat every insurgency by simply killing everyone. Also, in our model, not replacing civilians gives a huge advantage to the government, since no matter what, soldiers will eventually kill every insurgent (provided that *effectiveness* is not equal to 0). As we will shortly show, this is not the case when we use replacement. In fact, things get a lot harder for the government.
In Figures 8,9 we show the result with replacement, without allowing the civilians to move around the map. One can easily observe the discontinuity with respect to accuracy near the value 0.8, where for accuracy below this level, we get a large amount of increase in the average simulation time. To show how different parameters can heavily influence this result, we run further experiments, shown in Figures 10,11. There, we lower the *Influence Grid*, thus reducing the amount of citizens influenced by soldiers' counter attacks. Furthermore, we somewhat lowered the *fear increment* and *anger increment* values. In Table 11 we present the averaged values for the duration of simulations and total deaths.

Table 11: Results: Civilian replacement

	Average Time Steps	**Average Deaths**
Civilians can move	4188	1618
Civilians cannot move	4080	1609

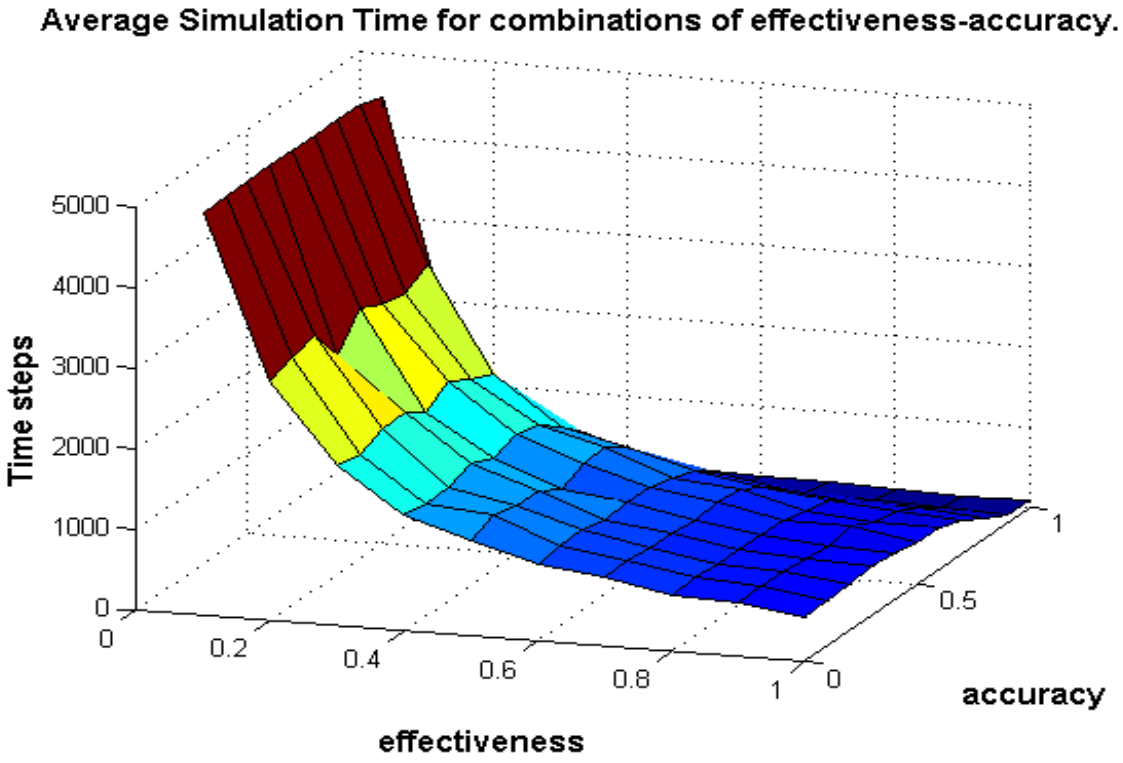

Figure 4: Average Simulation Time, for varying combinations of accuracy-effectiveness, with 10 simulations per combination. Parameter values are shown in Table 9. No civilian replacement. Civilians are not allowed to move around the map.

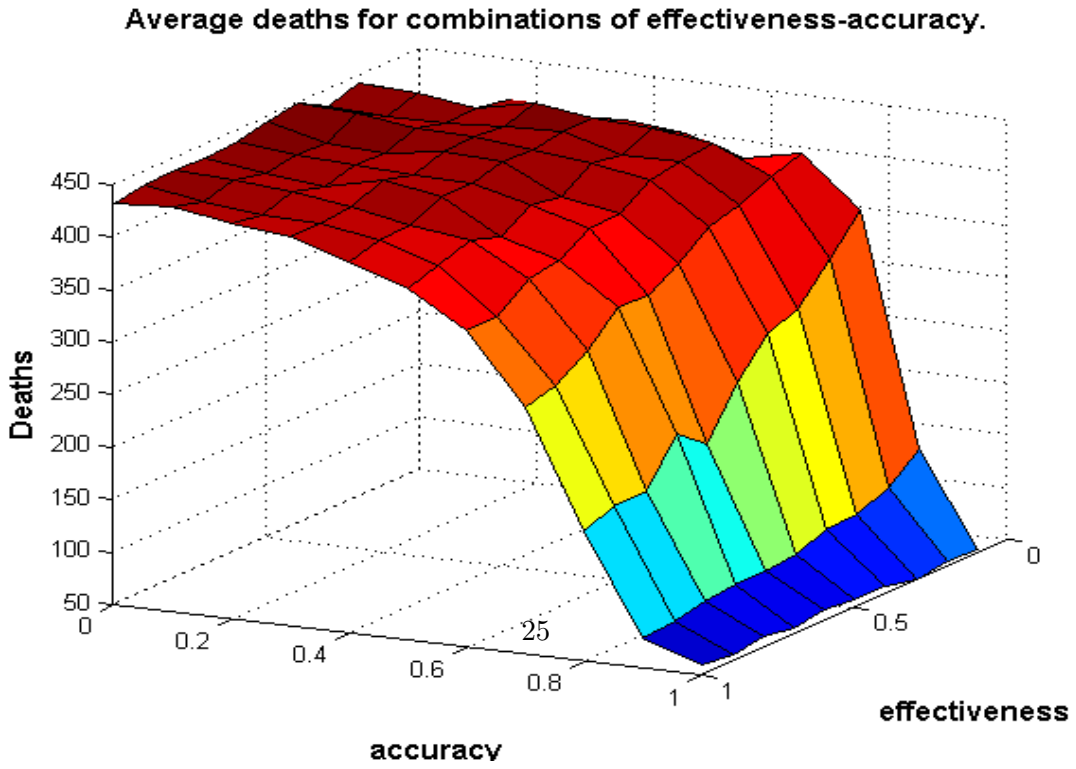

Figure 5: Average Deaths, for varying combinations of accuracy-effectiveness, with 10 simulations per combination. Parameter values are shown in Table 9. No civilian replacement. Civilians are not allowed to move around the map.

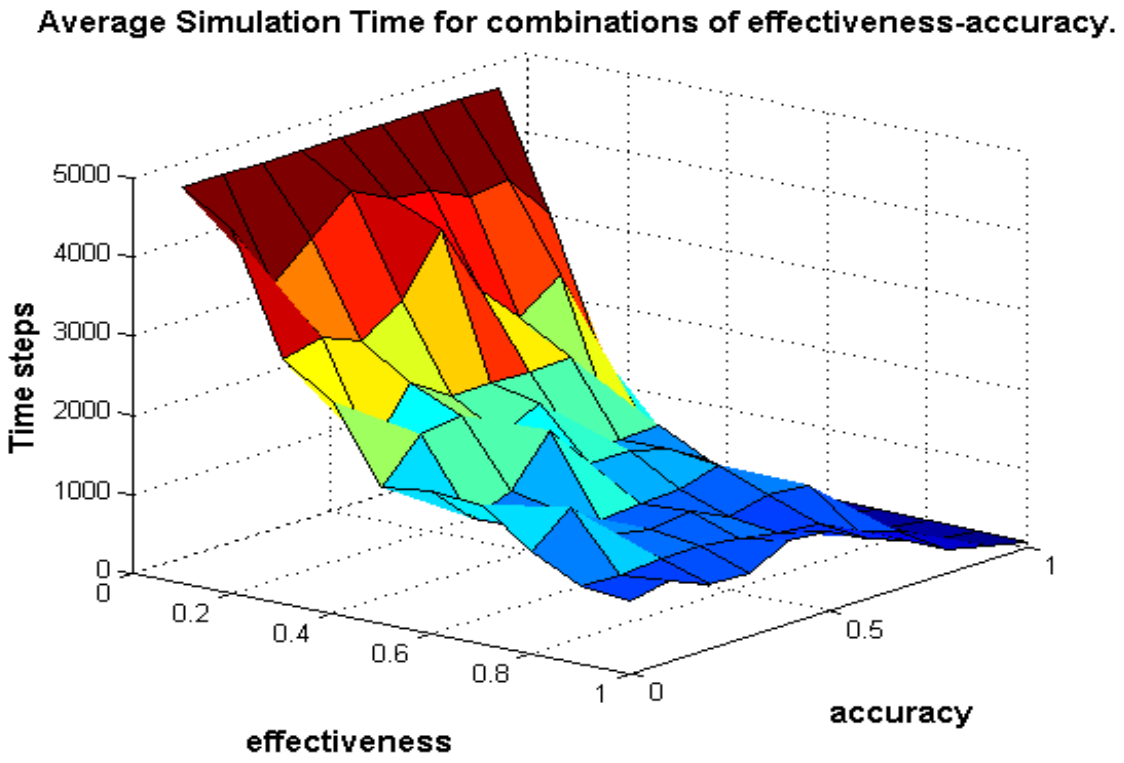

Figure 6: Average Simulation Time, for varying combinations of accuracy-effectiveness, with 10 simulations per combination. Parameter values are shown in Table 9. No civilian replacement. Civilians are allowed to move around the map.

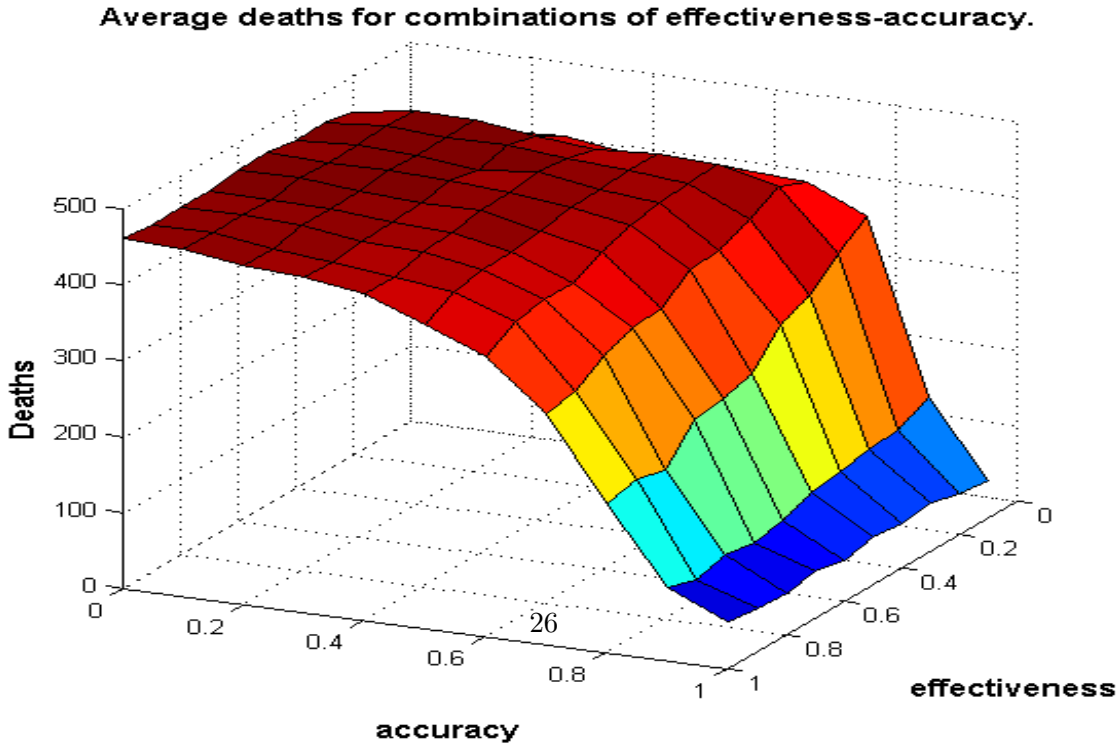

Figure 7: Average Deaths, for varying combinations of accuracy-effectiveness, with 10 simulations per combination. Parameter values are shown in Table 9. No civilian replacement. Civilians are allowed to move around the map.

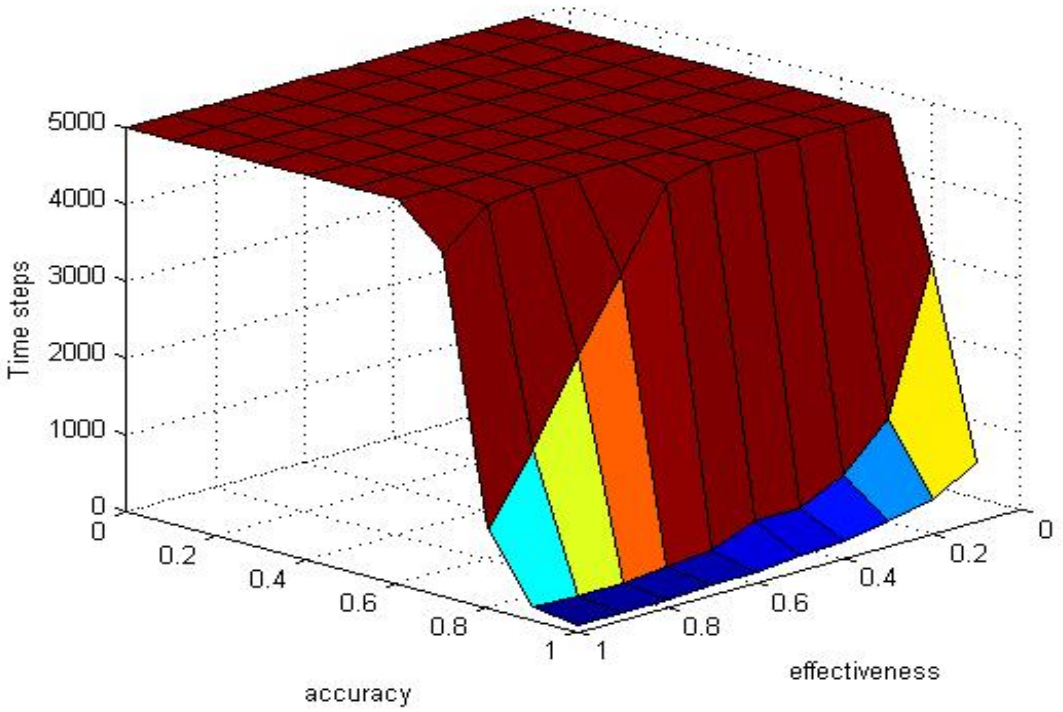

Figure 8: Average Simulation Time, for varying combinations of accuracy-effectiveness, with 10 simulations per combination. Parameter values are shown in Table 9. Civilian replacement. Civilians are not allowed to move around the map.

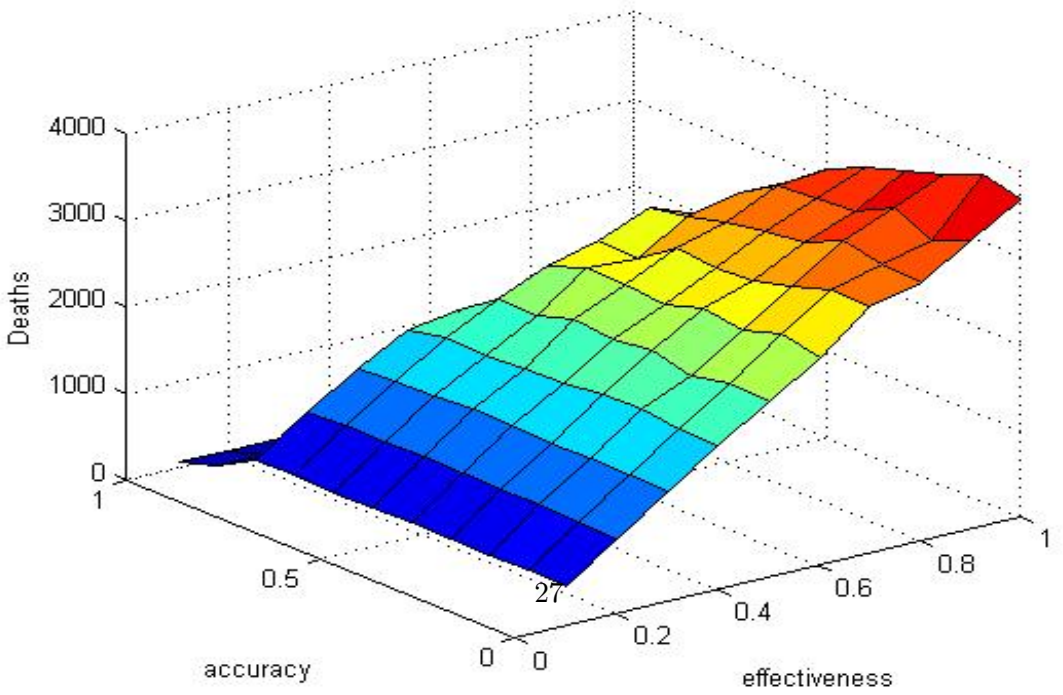

Figure 9: Average Deaths, for varying combinations of accuracy-effectiveness, with 10 simulations per combination. Parameter values are shown in Table 9. Civilian replacement. Civilians are not allowed to move around the map.

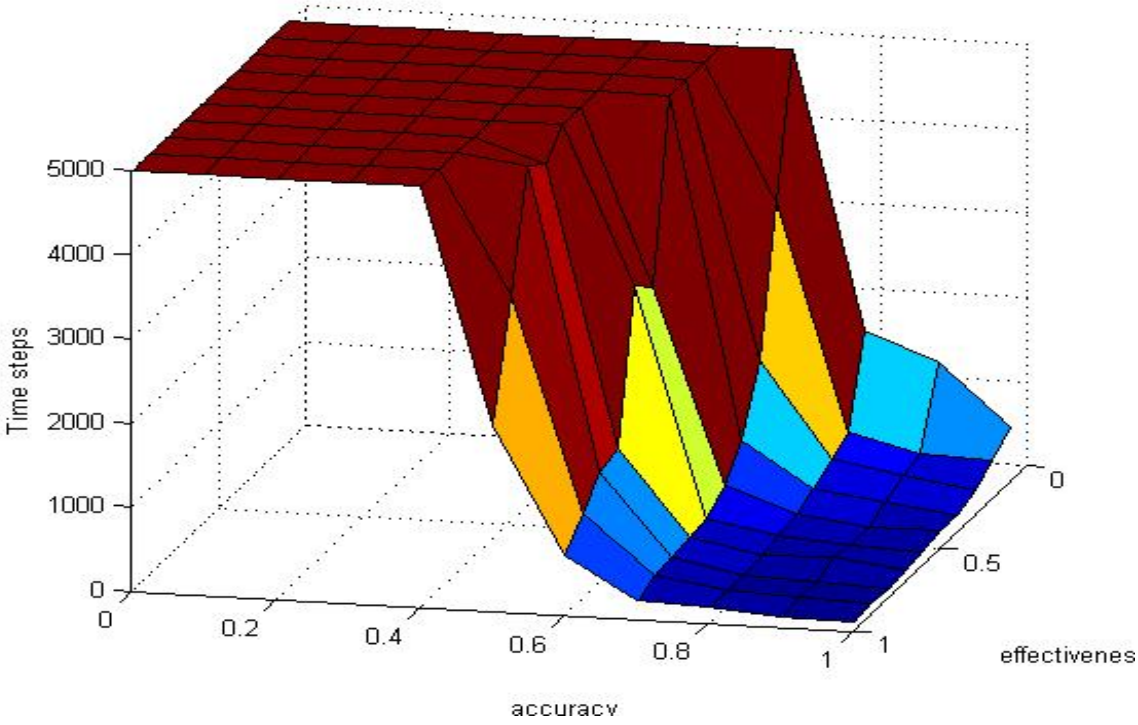

Figure 10: Average Simulation Time, for varying combinations of accuracy-effectiveness, with 10 simulations per combination. Attack Grid = Collateral Grid = Influence Grid = 9. Anger Increment = 0.02. All other parameters, follow Table 9.

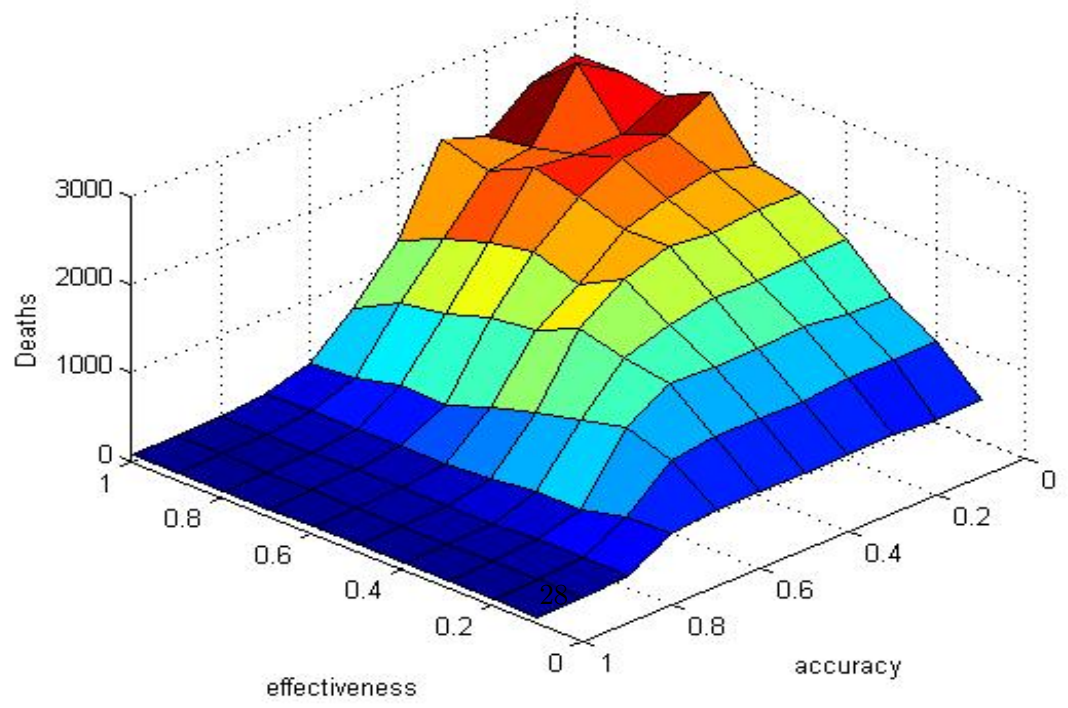

Figure 11: Average Deaths, for varying combinations of accuracy-effectiveness, with 10 simulations per combination. Attack Grid = Collateral Grid = Influence Grid = 9, Anger Increment = 0.02. All other parameters, follow Table 9.

A very important observation we obtain from Table 11 is the fact that when we replace dead civilians, we clearly extend the average duration of the insurgency. This is of course natural and anticipated, since the soldiers now have more work to do than simply killing insurgents. Replaced civilians may start with high values of anger and thus become latent insurgents right from the start. This situation is very unstable from a governmental point of view, and this can also explain why we have such a high sensitivity to accuracy. A second observation that when using replacement, the ability of civilians to move around the map in order to get closer to a soldier is irrelevant and does not influence the outcome of the insurgency. This is expected as well, because after all, when replacing dead civilians, there will (almost) always exist at least one of them within range of a soldier.

6.1.3 Individual Experiments

High effectiveness, high accuracy To illustrate the dynamics of our model and attempt to provide some intuitive explanation of what is really happening, we will present some individual experiments. The first two experiments deal with scenarios of high accuracy and effectiveness. Figures 12, 13 show the results when we do not replace dead civilians, whereas Figures 14, 15 are the results of an experiment with civilian replacement. First of all, what is easily observable is that both scenarios end with the insurgencies defeated. Hence, a quick remark would be, that when accuracy and effectiveness have high values ($\geq$0.8), the other parameters such as civilian replacement, are not very important. Hence, results with respect to accuracy and effectiveness are very robust. The main intuition behind this, is that when soldiers manage to kill the insurgents in the majority of counter attacks, whilst injuring very few civilians as well, then the model will eventually reach a minimum and the insurgency will be defeated; the only way for an insurgency to win, is by becoming self-sustaining, which means that soldiers actually help the insurgents by creating more insurgents than they can kill (via collateral damage). Of course in scenarios with such high accuracy and effectiveness ratings, this is not possible, and that is why in such cases, insurgencies are doomed to be defeated. Another thing we should point out, is that allowing replacement will in general (and in those particular experiments as well), increases the average duration of the insurgencies; without replacement we can see that the insurgency ended after about 200 steps, whilst with replacement, it ended after about 400 steps.

Let's see what happens when we lower either accuracy or effectiveness. In these scenarios, we consider civilian replacement, although useful results can be obtained even when we disable it.

High effectiveness,low accuracy We continue by simulating a scenario with high effectiveness (0.8) and low accuracy (0.2). This can be considered as a scenario where brute force (direct approach) and indiscriminant violence are employed by the government to defeat the rebels. What we can observe from Figure 17 is that there is a large number of deaths, disproportional to the initial size of the population. Notice also that the insurgency is not defeated; there is a large number of latent insurgents left on the map. An equivalent scenario, but with civilian replacement disabled, would have probably resulted to a defeated insurgency, even though most of the population would have died by then.

Low effectiveness,low accuracy Our next scenario, is what is considered in literature, a "nightmare" scenario for the government: namely ineffective use of brute force, resulting in many civilian casualties, while the main objective of killing the insurgents is not accomplished. Here, the results are more or less expected, especially in the more difficult scenario where dead civilians are replaced. There are two main things to notice from Figures 18, 19:

1. The insurgency has become self sustaining (in other words, we can assume that the rebels have won). The world at the end of the simulation is full of active and latent insurgents. This is natural since soldiers not only cannot kill the insurgents, but every time they try, they generate possibly even more of them.

2. The number of deaths is lower compared to the previous scenario. This is expected as well, since we have a low effectiveness rare, so few insurgents actually died. Notice that collateral damage cannot kill civilians; only injure them. This is of course a design decision, as it would be equally justified to assume that civilians can die from collateral damage (indeed some models make such assumptions).

High effectiveness,High accuracy, Leaders In order to illustrate the change in dynamics that we can obtain by adding leaders in our model, we close this section with a final experiment. For this purpose, we kept the same parameters as in our first experiment (high accuracy, high effectiveness, civilian replacement, civilians move around the map). The sizes of all grids are left the same. The results of the previous experiment are shown in Figures 14 and 15.Figures 20 and 21 summarize

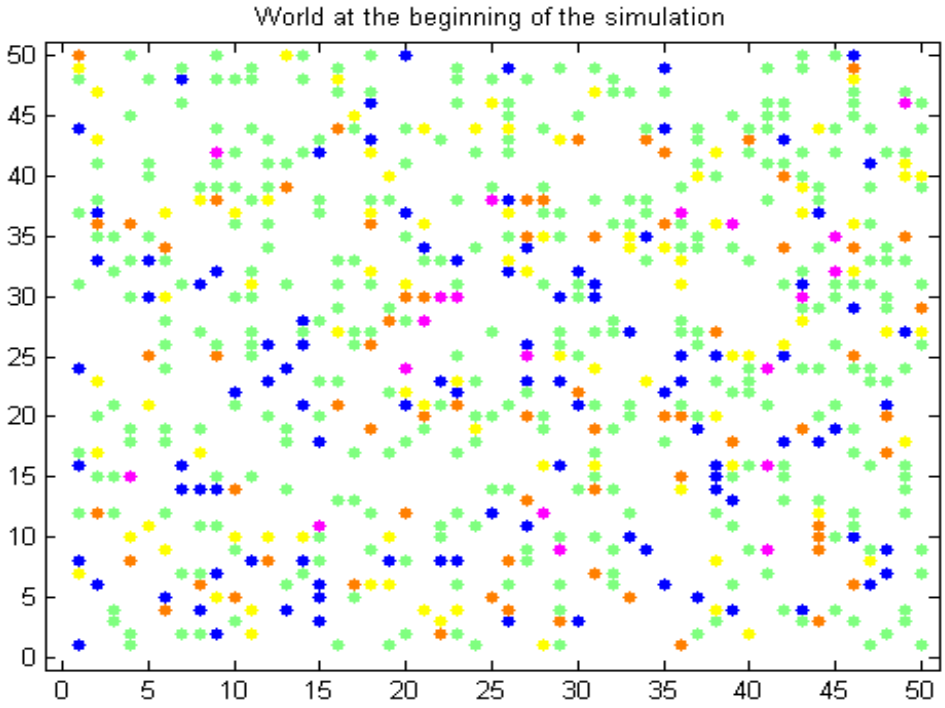

Figure 12: World at the beginning of an experiment using leaders, allowing civilians to move around the map, but having no replacement. Attack Grid = Collateral Grid =9, whereas Influence Grid = 12.

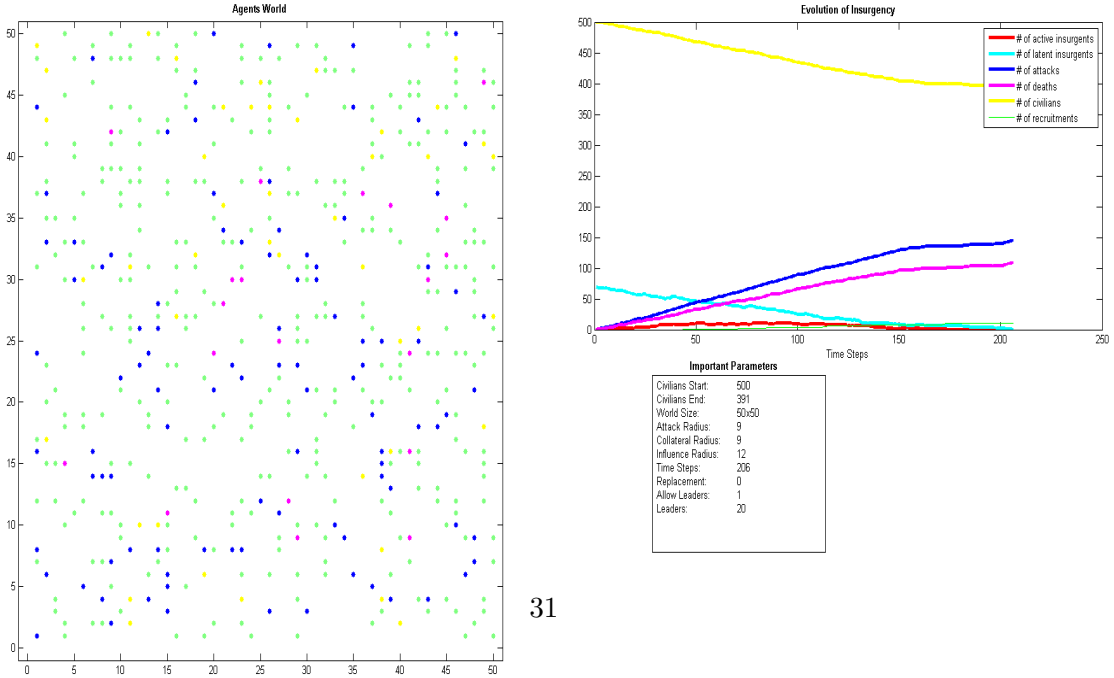

Figure 13: Results of the simulation.

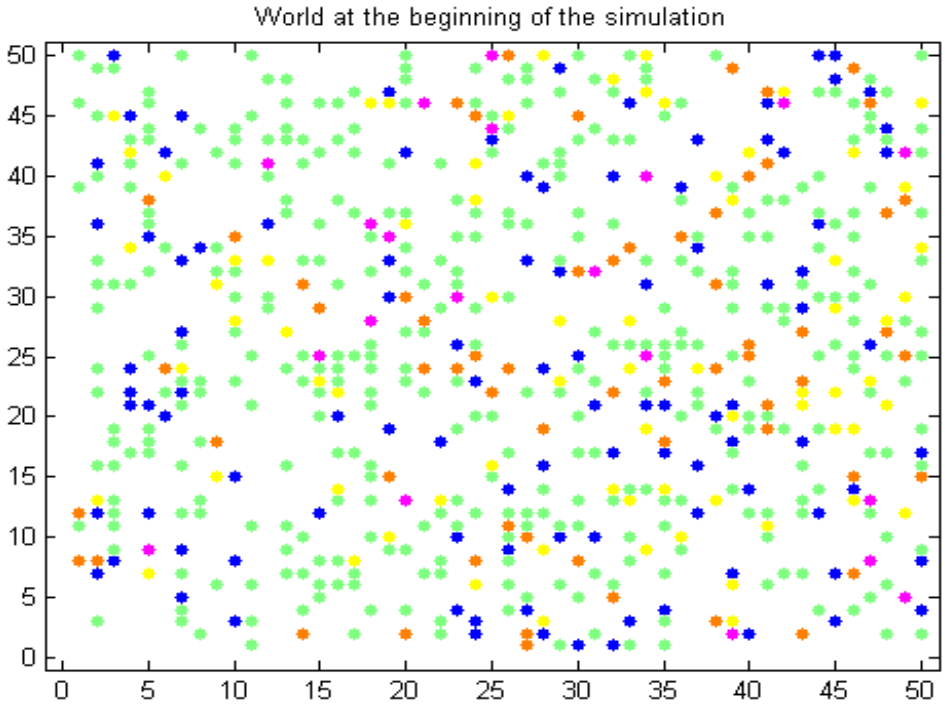

Figure 14: World at the beginning of an experiment, allowing civilians to move around the map, and replacement of dead civilians. Attack Grid = Collateral Grid =9, whereas Influence Grid = 12.

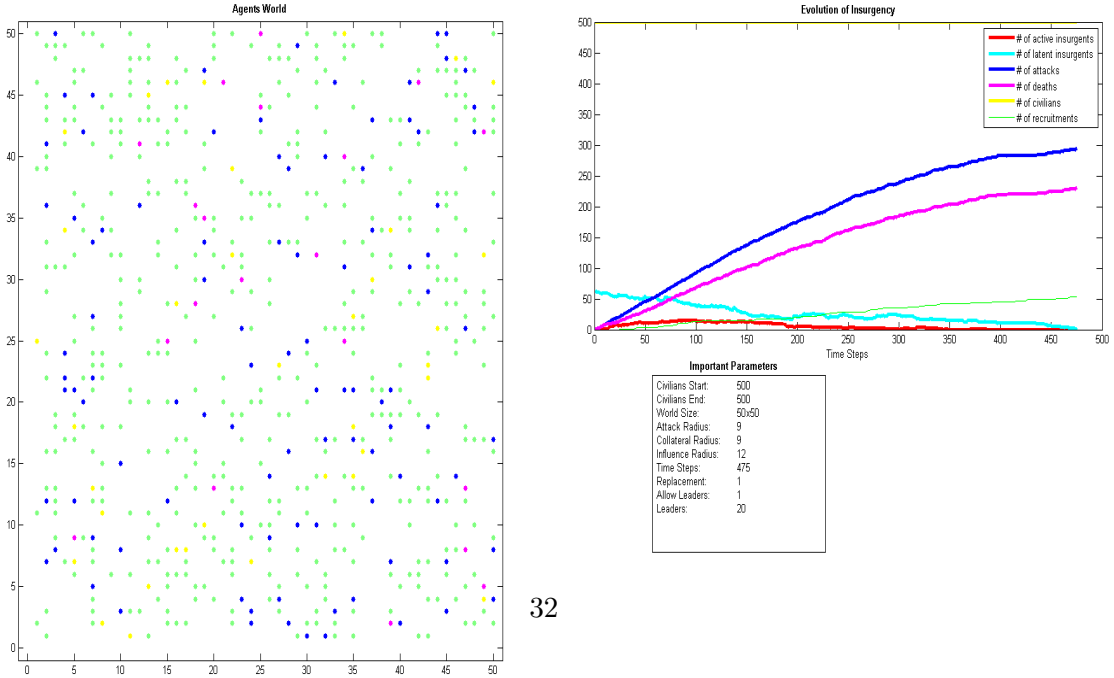

Figure 15: Results of the simulation.

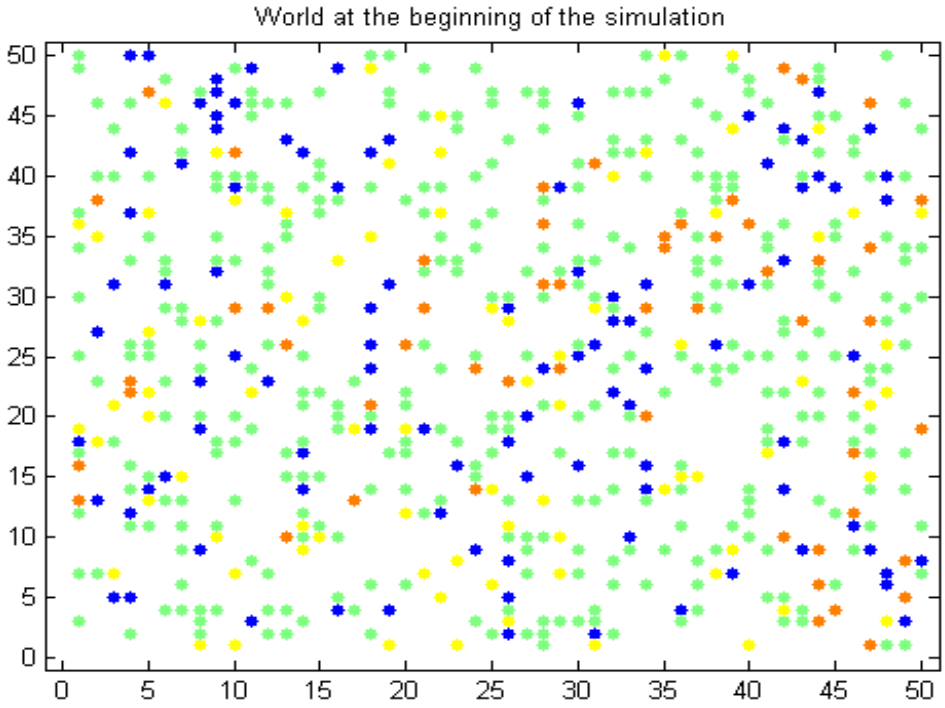

Figure 16: World at the beginning simulation: no leaders, civilians cannot move, dead civilians are replaced. Attack Grid = Collateral Grid =9, whereas Influence Grid = 15. Effectiveness = 0.8, Accuracy = 0.2.

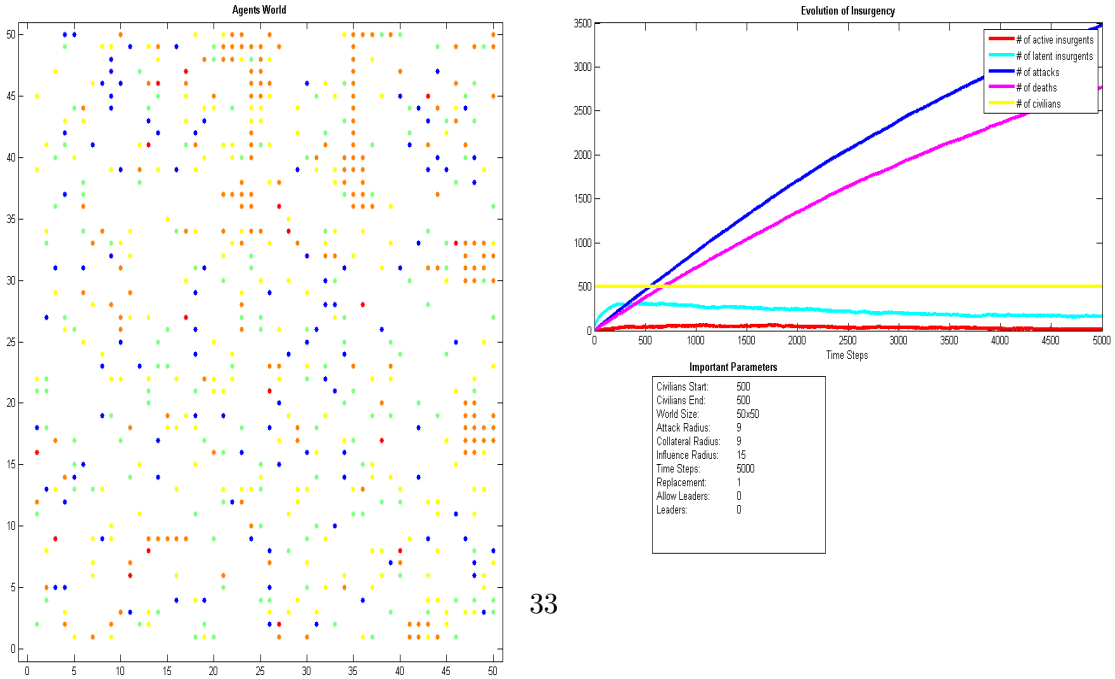

Figure 17: Results of the simulation.

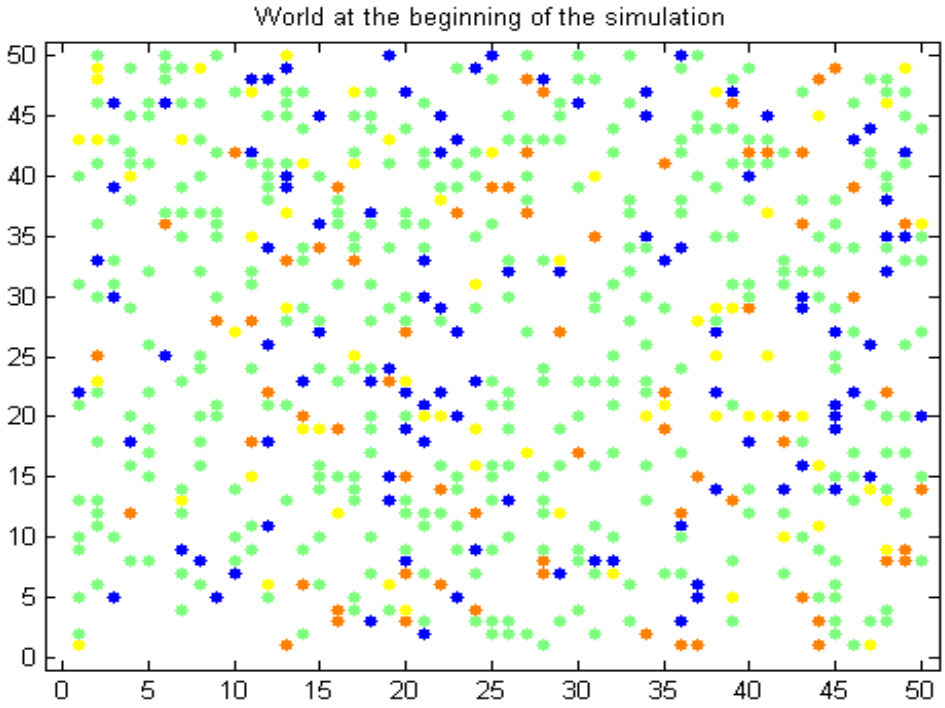

Figure 18: World at the beginning simulation: no leaders, civilians cannot move, dead civilians are replaced. Attack Grid = Collateral Grid =9, whereas Influence Grid = 15. Effectiveness = 0.2, Accuracy = 0.2.

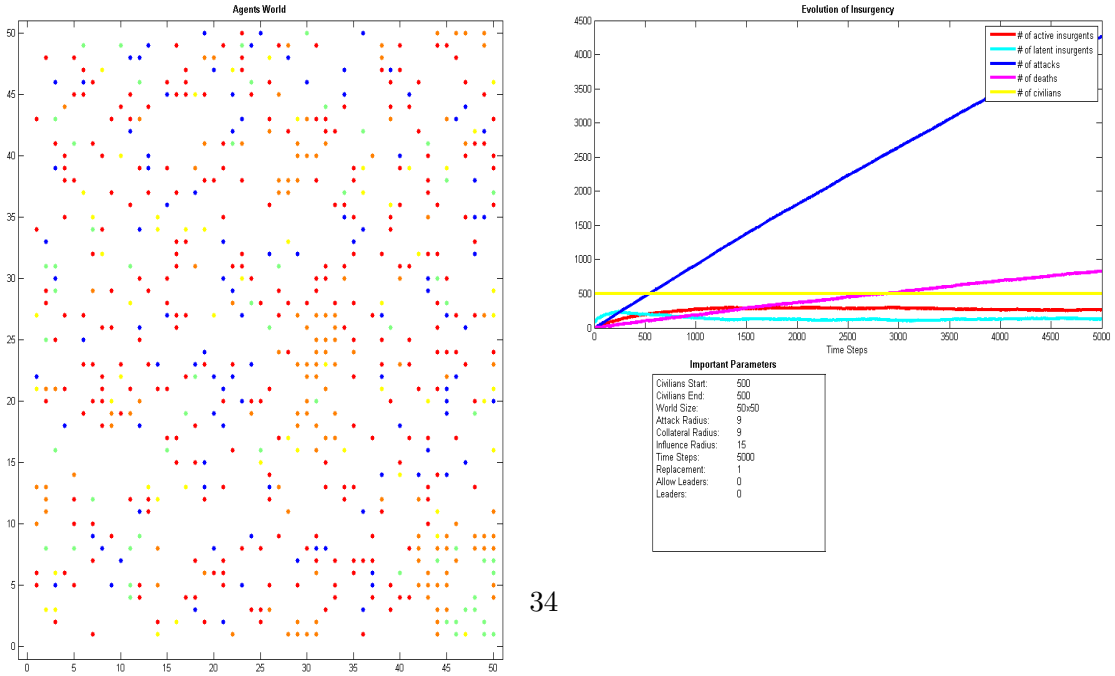

Figure 19: Results of the simulation.

the result of the current experiment. The main thing we should note here, is that the soldiers need much more time to defeat the insurgency: recall that in the previous experiment with the same parameters but without leaders, the simulation ended after 400 steps, whereas the current experiment ended after 752 steps. The parameters used for the leaders are shown in Table 12.

Table 12: Leader Parameters for experiment

Parameter	**Mean**	**Standard Deviation**
L.influenceChance	0.2	0.001
L.recruitChance	0.05	0.001
L.influenceRange	20	2
L.influenceRate	0.05	0.001

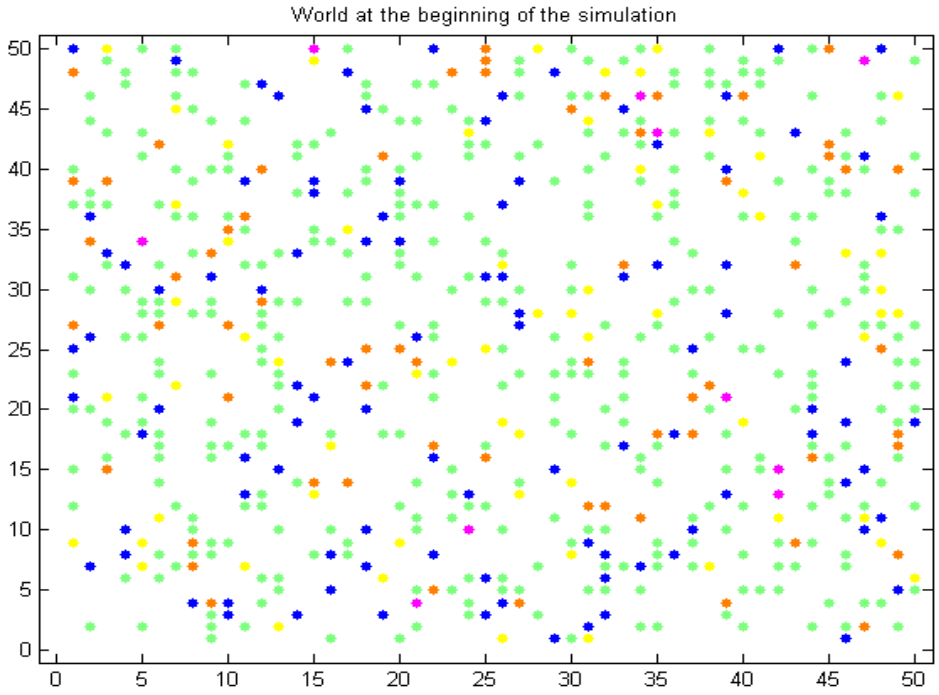

Figure 20: World at the beginning simulation: leaders included, civilians are allowed to move, dead civilians are replaced. Attack Grid = Collateral Grid =9, whereas Influence Grid = 12. Effectiveness = 0.8, Accuracy = 0.8.

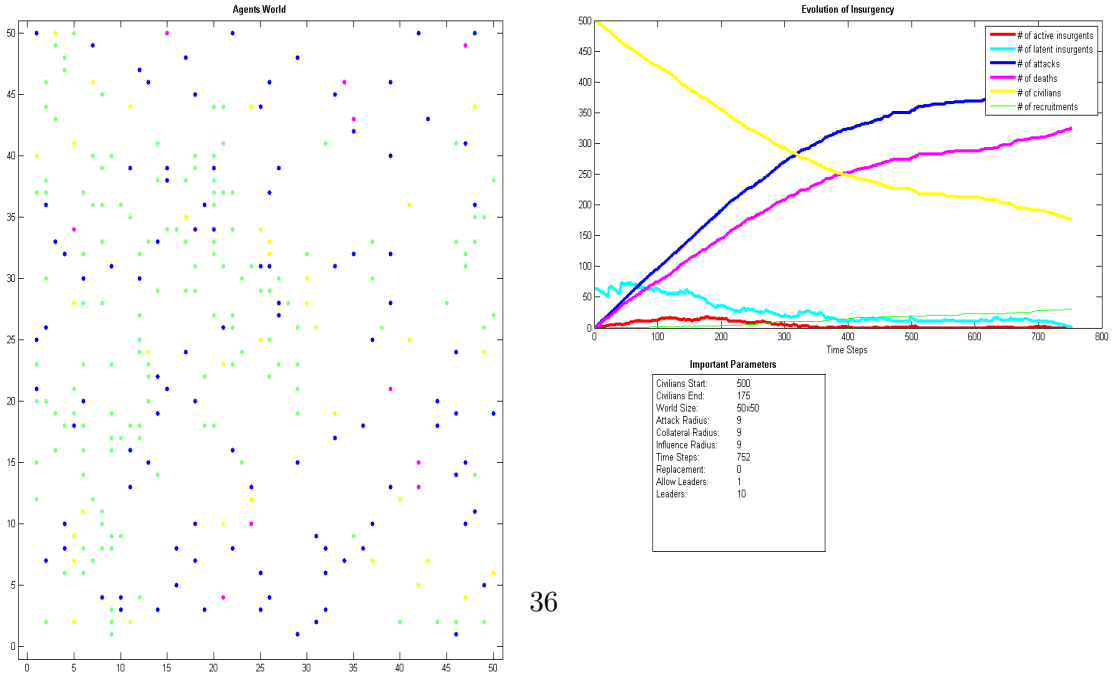

Figure 21: Results of the simulation.

6.2 Discussion

We have already discussed most of the results obtained from the simulations. Here we are going to summarize them, and discuss how they relate to real world scenarios.

1. To begin with, recall the scenarios without replacement. The results obtained from Figures 4, 6 show something very important, yet intuitive from a theoretical point of view. In such scenarios it doesn't matter whether the government uses a direct or indirect approach to counter the insurgency. Only the ability of the soldiers in neutralizing the insurgents matters. In other words, there is theoretically no reason to try and avoid collateral damage, civilian injuries etc, since sooner or later, the insurgents will be eliminated. This is seen in the aforementioned Figures, were we see that the average duration of the insurgency depends mainly on the *effectiveness*, whilst been very robust to changes in *accuracy*.

2. The situation is a bit different with regard to the total deaths during an insurgency. Notice from Figures 5, 7, that there is a different relation between accuracy, effectiveness and the number of casualties. Specifically, in such scenarios, deaths depend mainly on *accuracy* and are very robust to changes in *effectiveness*. This may sound weird but in fact, if we think about it, it tells us that if we have low accuracy, we actually create more insurgents that eventually will be killed afterwards by the soldiers. Increasing accuracy, will result in less new insurgents and hence a lower number of total deaths. On the other hand, increasing or decreasing effectiveness does not influence deaths (at least not so much), because as we said, those insurgents will sooner or later be neutralized.

3. The remarkable observation is that the situation is completely reversed in scenarios with civilian replacement. In this case, as shown in Figures 8, 9, the duration of an insurgency is dependant only on *accuracy*. Furthermore, there is an accuracy threshold, which depends on the model parameters (compare with Figures 10, 11); dropping below this threshold usually results in long-term insurgencies, which usually become self-sustaining and end with the rebels being victorious. Furthermore, this threshold depends on how soft or hard are our model parameters: small interaction grids, low fear and anger increments etc, usually lower this threshold. Notice that effectiveness is barely influencing the outcome.

These conclusions we just discussed can be directly extended to real world situations. When we discussed at the introduction of this project, the two main approaches of modern counter insurgency literature, we mentioned two main doctrines: *direct approach*, which supports the focus on high effectiveness and does not really care

about accuracy, and *indirect approach*, which supports exactly the opposite. Now what we can deduce from the former analysis, is that overall, on average, the *indirect approach* is much more effective at dealing with insurgencies. Hence, government forces should focus on tactics, weapons and training, that allows them to lower civilian casualties as much as possible. Notice however that there is a catch: this conclusion is based on several assumptions, the most important one being that we are considering a relatively densely populated world. This means that civilian are closely packed, or that there at least a few civilian clusters (possibly representing cities or towns). In such a case, then *yes*, accuracy is the key. However, there might be other cases, for example with sparsely populated areas, where accuracy would not be as important. Indeed it such a case, one should focus on neutralizing the insurgents, and not really care about collateral damage, since by default there cannot be too much of it.
This represents another example of how complicated is the issue we are dealing with, and how many factors should be taken into account in order to accurately simulate it.

7 Summary and Outlook

In this project, we have studied and extended Bennett's model on *modelling the early dynamics of insurgencies*. We mainly focused on improving the model by making it more realistic and configurable so as to broaden its application to different circumstances. For instance, lets say that we want to gain insight into the evolution of an insurgency in a specific country. To do so, we can simply provide our model with a number of parameters that characterize this country (distribution on wealth, population density, etc) and we will get a result that is much closer to reality than the general purpose model. Of course we can not imply that these results could represent real facts, but they may be seen as possible outcomes and should be taken into account when governments decide their policy towards insurgents.

Moreover, we managed to verify all the conclusions of Bennett, since his work can now be thought of as a special case of our wider model. The most important fact that we wanted to prove is that accuracy is the most critical parameter in defeating long-term insurgencies under some mild assumptions. Thus, reducing civilian casualties when dealing with an insurgency, is far more important than using a brute force strategy against the rebels. Our increased parametrization of the model, might have led to more complex agents and action-effects but on the other hand we have added much more degrees of freedom and an interested researcher could use these features, or even add more, to further investigate the dynamics of an insurgency.

8 References

BENNETT, D. Scott (2008), 'Governments, Civilians, and the Evolution of Insurgency: Modeling the Early Dynamics of Insurgencies ', *Journal of Artificial Societies and Social Simulation,* Volume 11, No. 4 7.

9 Appendix

In this section, we present the code we wrote for the simulations.

9.1 Main

```
1 %% Soldiers
2 %
3 % Global Variables:
4 %
5 % effectiveness :=  (probability to kill an insurgent)
```

```
% accuracy      :=  (1—accuracy = chance to injure neighbouring civilians)
%
% 100 total soldiers

%% Civilians
%
% Individual characteristics, ranging between 0 and 1:
%
% 1. Anger(0 = not angry at all,1 = maximum anger).
% 2. Fear(0 = not afraid, 1 = maximum fear).
% 3. Violence Threshold(0 =  low threshold, easy to turn to violence, 1 ...
    = nearly impossible to become an insurgent).
%
% LATENT INSURGENT = (Anger > Fear && Anger>Violence Threshold).
% 500 total civilians.
% Initial values are drawn from a normal distribution.
%
%                          Mean        Standard Deviation
% Anger                    0.25        0.125
% Fear                     0.5         0.25
% Violence Threshold       0.5         0.25

%% Map
%
% 2D, with edges.
% Only neighbours can interact.
% gridSize x gridSize (typically 50x50)

%% Map Display
%
% Soldiers = blue.
% Civilians {green, yellow, orange, red}.
%
%    green  = anger<fear && anger<violence threshold.
%    yellow = anger>fear && anger<violence threshold.
%    orange =
%    red    =

%% Start

clear all
close all
clc

global Civilians
global Leaders
global Soldiers
global noc
global param

```

```
55 param = parameters();
56
57 %%  Perform single simulation
58 if ¬param.batch
59
60     Results = completeSimulation();
61     Results.status
62     figure1 = displayResults(Results);
63     save('parameters','param');
64     save('results','Results');
65
66 else
67
68     %% Perform repetitive simulations
69
70     [X,Y] = meshgrid((1:10)'/10,(0:10)'/10);
71
72     simulations = zeros(size(X));
73     count       = 0;
74
75     %   Do multiple simulations for all combinations of effectiveness and
76     %   accuracy, with/without switchPosition.
77
78     for afterlife = 0:1
79
80         param.afterlife = afterlife;
81
82         for switchPosition=0:1
83
84             param.switchPosition = switchPosition;
85
86             for eff = 1:10
87
88                 effectiveness = eff/10;
89
90                 for acc = 0:10
91
92                     accuracy = acc/10;
93                     SumOfTimeSteps = 0;
94
95                     for i = 1:param.runs
96
97                         nos = param.nos;
98                         noc = param.noc;
99
100                        param.effectiveness = effectiveness;
101                        param.accuracy      = accuracy;
102
103                        Result = completeSimulation();
104
```

```
105                         %   Collect results for this simulation.
106                         Results(eff,acc+1,i) = Result;
107
108                         SumOfTimeSteps = SumOfTimeSteps + Result.timeSteps;
109
110                     end
111
112                     timeSteps = SumOfTimeSteps/param.runs;
113
114                     position = (X == eff/10).*(Y == acc/10);
115                     index    = find(position == 1);
116                     [r c]    = ind2sub(size(X),index);
117
118                     simulations(r,c) = timeSteps;
119
120                     count = count + 1;
121                     fprintf('Simulation number:%d has ended in %d ...
                            steps\n',count,timeSteps)
122
123                 end
124
125             end
126
127
128         end
129     end
130
131 end
```

9.2 Functions

```
1  function [C] = civilian(x,y)
2
3  %CIVILAN    Generates an agent of type 'civilian' and returns its struct.
4  %
5
6  %%  Parameters of gaussian distributions:
7
8  %   Rich
9
10 mean1a = 0.15;
11 std1a  = 0.125^2; %0.125^2
12 mean1f = 0.7;
13 std1f  = 0.25^2;
14 mean1t = 0.8;
15 std1t  = 0.25^2;
16
```

```
% Middle

mean2a = 0.25;
std2a  = 0.125^2;
mean2f = 0.5;
std2f  = 0.25^2;
mean2t = 0.5;
std2t  = 0.25^2;

% Poor

mean3a = 0.45; %0.45
std3a  = 0.125^2;
mean3f = 0.25;
std3f  = 0.25^2;
mean3t = 0.4;
std3t  = 0.25^2;

%% Creating civilian

dice = 100*rand(1);

if dice > 85

    C.wealth = 'Rich';
    C.anger  = gsample(mean1a,std1a,1);
    C.fear   = gsample(mean1f,std1f,1);
    C.thres  = gsample(mean1t,std1t,1);

elseif dice > 20

    C.wealth = 'Middle';
    C.anger  = gsample(mean2a,std2a,1);
    C.fear   = gsample(mean2f,std2f,1);
    C.thres  = gsample(mean2t,std2t,1);


else

    C.wealth = 'Poor';
    C.anger  = gsample(mean3a,std3a,1);
    C.fear   = gsample(mean3f,std3f,1);
    C.thres  = gsample(mean3t,std3t,1);


end

C.x = x;
C.y = y;

```

```
67 %% Limits [0,1]
68
69 C.anger(C.anger < 0) = 0;
70 C.fear(C.fear   < 0) = 0;
71 C.thres(C.thres < 0) = 0;
72
73 C.anger(C.anger > 1) = 1;
74 C.fear(C.fear   > 1) = 1;
75 C.thres(C.thres > 1) = 1;
76
77 C.attack = 0;               %   attack = 1 => civilian has made an attack.
78 C.threat = 0;               %   threat = 1 => civilian is at least latent.
79
80 C.Colour = [ 0 0 0];
81
82 end
```

```
1  function [S] = soldier(x,y)
2
3  %SOLDIER    Generates an agent of type 'soldier' and returns its struct.
4
5  %% Parameters of gaussian distributions:
6  global param
7
8  effectiveness = param.effectiveness;
9  accuracy      = param.accuracy;
10 sensitivity   = param.sensitivity;
11
12 stde = 0;
13 stda = 0;
14
15 S.effectiveness = gsample(effectiveness, stde, 1);
16 S.accuracy      = gsample(accuracy,      stda, 1);
17
18 S.effectiveness(S.effectiveness < 0) = 0;
19 S.accuracy(S.accuracy < 0) = 0;
20 S.effectiveness(S.effectiveness > 1) = 1;
21 S.accuracy(S.accuracy >1) = 1;
22
23 S.sensitivity = sensitivity;
24
25 S.x = x;
26 S.y = y;
27 S.Colour = [0 0 1];
28
29 end
```

```
function L = leader(x,y)

%Leader     This function creates an agent of type 'leader'. These are
%           special types of agents that do not fight, but have some ...
    special roles,
%           such as recruiting and helping spread the insurgency.

%   infuence, power
%   influenceChance is the probability that this leader will perform a
%   propaganda action.
%   influenceRange is the range of all civilians affected by the propaganda
%   action.
%   influenceRate is the impact that this propaganda will have an all
%   affected civilians.
%   recruitChance, is the probability that this leader will succesfully ...
    recruit a
%   civilian.
global gridSize

influenceChance_mean = 0.4;
influenceChance_std  = 0.001;

recruitChance_mean     = 0.1;
recruitChance_std      = 0.001;

influenceRange_mean = 20;
influenceRange_std  =  2;

influenceRate_mean  = 0.1;
influenceRate_std   = 0.001;

L.influenceChance = gsample(influenceChance_mean,influenceChance_std,1);
L.recruitChance   = gsample(recruitChance_mean,recruitChance_std,1);
L.influenceRange  = ...
    round(gsample(influenceRange_mean,influenceRange_std,1));
L.influenceRate   = gsample(influenceRate_mean,influenceRate_std,1);

if L.influenceRate>1

    L.influenceRate = 1;

elseif L.influenceRate<0

    L.influenceRate = 0;

end

if L.influenceRange>gridSize

```

```
47      L.influenceRange = gridSize;
48
49  elseif L.influenceRange<0
50
51      L.influenceRange = 0;
52
53  end
54
55  if L.influenceChance>1
56
57      L.influenceChance = 1;
58
59  elseif L.influenceChance<0
60
61      L.influenceChance = 0;
62
63  end
64
65  if L.recruitChance>1
66
67      L.recruitChance = 1;
68
69  elseif L.recruitChance<0
70
71      L.recruitChance = 0;
72
73  end
74
75  L.x = x;
76  L.y = y;
77  L.Colour = [1 0 1];
78
79  end
```

```
1   function   Results = completeSimulation()
2
3   % COMPLETESIMULATION    This function performs a complete simulation. It
4   %                       will return all the information required for the
5   %                       post-processing.
6   %
7   % INPUT                 If the variable disp is set to 1 the visualization
8   %                       of the world will be active.
9   %
10  % OUTPUTS:              active -> number of active insurgents at any given
11  %                       time step.
12  %
13  %                       latent -> number of latent insurgents at any given
14  %                       time step.
```

```
15 %
16 %                       civilians -> number of civilians at any given time
17 %                       step (use when replacement = 0).
18 %
19 %                       deaths -> number of civilian casualties at any
20 %                       given time step.
21 %
22 %                       attacks -> number of attacks carried out by the end
23 %                       of any given time step.
24 %
25 %                       timeSteps -> duration of the simulation
26 %
27 %                       status -> why simulation terminated?
28 %                                   -   defeated: no latent insurgents ...
       remain
29 %                                   -   won: number of casualties ...
       exceeded a certain threshold
30 %                                   -   undecided: no possibility of ...
       further attacks
31 %                                   -   expired: reached maxSteps
32
33 global world
34
35 global Civilians
36 global Soldiers
37 global Leaders
38
39 global param
40 global noc
41 global vidObj
42 global h
43
44 noc = param.noc;
45 %   Initialize the world
46 world = createWorld();
47
48 %   Populate the world, according to the initialization
49 %   Assign coordinates to human structures
50 [Civilians,Soldiers,Leaders] = populateWorld();
51
52 updateWorld();
53
54 if ¬param.batch
55
56     h = figure;
57     displayWorld;
58     title('World at the beginning of the simulation')
59
60 end
61
```

```
%   Initialize matrices to store important information regarding the
%   evolution of the insurgency.
maxStep = param.maxStep;

attacks = zeros(1,maxStep);     %   # of attacks that have occured
active = zeros(1,maxStep);      %   # of active insurgents
latent = zeros(1,maxStep);      %   # of latent insurgents
civilians = zeros(1,maxStep);   %   # of civilians alive
deaths = zeros(1,maxStep);      %   # of deaths
recruits = [];                  %   # of succesful recruitments

if param.allowLeaders

    recruits = zeros(1,maxStep);

end

active(1)           = length(find(world(:,:,1)==6));
latent(1)           = length(find(world(:,:,1)==5));
civilians(1)        = param.noc;

status = 'expired';

if param.video

    vidObj = VideoWriter('video.avi');
    open(vidObj);

end

%   Repeat simulation step, until either a terminal condition has been
%   reached, or the maximum number of iterations has been attained.
for i=2:maxStep

    timeSteps = i;

    %   Simulate
    [attackFlag,deathFlag,recruitFlag] = simulate2();

    %   Store the results for the i-th step
    active(i)       = length(find(world(:,:,1)==6));
    latent(i)       = length(find(world(:,:,1)==5));
%     civilians(i)    = length(find(world(:,:,1)>=3));
    civilians(i) = noc;
    attacks(i)  = attacks(i-1)+attackFlag;
    deaths(i)   = deaths(i-1)+deathFlag;

    if param.allowLeaders == 1

        recruits(i) = recruits(i-1) + recruitFlag;
```

```
112
113     end
114
115     %   Check for terminal conditions.
116     %   (a) All insurgents are dead.
117     %   (b) A certain number of deaths has occured.
118     %   (c) All insurgents are out of range of soldiers (and position
119     %       switching is not allowed).
120
121     if latent(i) == 0 && active(i) == 0
122
123         status = 'defeated';
124         break
125
126     end
127
128 %     if deaths(i)> param.percent*param.noc
129 %         status = 'won';
130 %         break
131 %     end
132
133     if outOfRange
134
135         switch param.switchPosition
136
137             %   If no position switching is allowed, then if all insurgents
138             %   are out of range, end the simulation.
139             case 0
140
141                 status = 'undecided';
142                 break
143
144             %   Else, choose a random insurgent and move him close to a
145             %   soldier so that simulation can continue.
146             case 1
147
148                 ind  = find(world(:,:,1)≥5);
149
150                 if param.allowLeaders == 1
151
152                     indLeaders = find(world(:,:,1)==7);
153                     ind = setdiff(ind,indLeaders);
154
155                 end
156
157                 [row col] = ind2sub(size(world(:,:,2)),ind);
158                 threats = [row col];
159                 pick = randi([1 size(threats,1)],1,'uint32');
160                 civilianToBeMoved = ...
                        world(threats(pick,1),threats(pick,2),2);
```

```
                moveInsurgent(civilianToBeMoved);
                updateWorld();

            otherwise

        end

    end

    if param.video
        clf(h,'reset')
        displayWorld();
        currFrame =getframe(h);
        writeVideo(vidObj,currFrame);

    end

end

%
% if ¬param.batch
%
%     displayWorld();
%     title('World at the end of simulation')
% end

%   Return the results of the simulation in a struct.
if ¬param.video
    Results = ...
        struct('active',active,'latent',latent,'civilians',civilians,'deaths',deaths,'attacks'
    attacks,'recruits',recruits,'timeSteps',timeSteps,'status',status);
else
    Results = ...
        struct('active',active,'latent',latent,'civilians',civilians,'deaths',deaths,'attacks'
    attacks,'recruits',recruits,'timeSteps',timeSteps,'status',status,'video',vidObj);

end
```

```


function [attackFlag,deathFlag,recruitFlag] = simulate2()

%SIMULATE   This function performs one simulation step.

global param
global noc
global Civilians
global Soldiers
```

```
11 global Leaders
12 global world
13 global vidObj
14 global h
15 nol = param.nol;
16
17 AttackGrid      = param.AttackGrid;
18 CollateralGrid  = param.CollateralGrid;
19 InfluenceGrid   = param.InfluenceGrid;
20 afterlife       = param.afterlife;
21 allowLeaders    = param.allowLeaders;
22 example         = param.example;
23 video           = param.video;
24
25 attackFlag  = 0;                                % In order to count the ...
       number of attacks
26 deathFlag   = 0;
27 recruitFlag = 0;
28
29 %   If leaders are allowed..
30 if allowLeaders == 1
31
32     %   pick a random leader
33     leader = randi([1 nol],1,'uint32');
34
35     influence_roll = rand;
36     recruit_roll   = rand;
37
38     %   If roll succesful, then perform a propaganda action.
39     if influence_roll < Leaders(leader).influenceChance
40
41         civ = ...
               findAgents(Leaders(leader).x,Leaders(leader).y,'civilian',Leaders(leader).influenc
42         propaganda(civ,Leaders(leader).influenceRate)
43
44     end
45
46     %   If roll succesful, then perform a recruit action.
47     if recruit_roll < Leaders(leader).recruitChance
48
49         Civilians   = recruit(Civilians);
50         recruitFlag = 1;
51
52     end
53
54 end
55
56 player = randi([1 noc],1,'uint32');
57
58 % Continue searching for a civilian that can make an attack.
```

```
59 while Civilians(player).threat == 0
60
61     player = randi([1 noc],1,'uint32');
62
63 end
64
65 x = Civilians(player).x;
66 y = Civilians(player).y;
67
68 %   Find all soldiers near the civilian
69 nearbySoldiers = findAgents(x,y,'soldier',AttackGrid);
70
71 if ¬isempty(nearbySoldiers)
72
73     %   Pick a random soldier nearby
74     victimindex = randi([1 length(nearbySoldiers)],1,'uint32');
75
76     victim = nearbySoldiers(victimindex);
77
78     Civilians(player).attack = 1;
79
80     attackFlag = 1;
81
82     world(x,y,1) = 6;
83
84     response = rand;
85
86     if response ≤ Soldiers(victim).sensitivity  % then . . . ATTACK!
87
88         %         display(' A T T A C K !');
89         %         fprintf('\n Insurgent in position (%d,%d) is being ...
               attacked !\n',x,y);
90
91         %         kill = rand;
92         %
93         %         if kill < Soldiers(victim).effectiveness
94         %
95         %             deathFlag = 1;
96         %             afterLife(player,afterlife); % REGENERATION OR NOT
97         %
98         %         end
99
100        %   Counterattack... find civilians close to the insurgent who
101        %   attacked.
102        nearbyCivilians = findAgents(x,y,'civilian',CollateralGrid);
103
104        damage = 0;
105
106        if ¬isempty(nearbyCivilians)
107            nearbyCivilians(2,:) = 0;
```

```
108         end
109
110         % Here we store if the civilian was injured by the attack
111         for k = 1:size(nearbyCivilians,2)
112
113             injury = rand;
114
115             if injury < 1 - Soldiers(victim).accuracy
116
117                 damage = damage+1;
118                 nearbyCivilians(2,k) = 1;
119
120             end
121
122
123         end
124
125         % UPDATE INJURED CIVILIANS AND SPECTATORS
126
127         influencedCivilians = findAgents(x,y,'civilian',InfluenceGrid);
128
129
130         if ¬isempty(influencedCivilians)
131
132             influencedCivilians(2,:) = 0;
133             for i=1:size(nearbyCivilians,2)
134
135                 if nearbyCivilians(2,i)==1
136                     influencedCivilians(2,influencedCivilians(1,:)==nearbyCivilians(1,i)) ...
                            = 1;
137                 end
138
139             end
140
141             for k = 1:size(nearbyCivilians,2)
142
143                 updateCivilian(nearbyCivilians(:,k),damage);
144
145             end
146
147         end
148
149         %   For displaying purposes only!!
150         if example==1
151
152             displayWorld([player victim], nearbyCivilians);
153             pause
154
155         end
156
```

```
157         if video
158
159             clf(h,'reset')
160             updateWorld();
161             displayWorld();
162             currFrame =getframe(h);
163             writeVideo(vidObj,currFrame);
164
165         end
166         %   Roll for whether the insurgent who attacked, will die.
167         kill = rand;
168
169         if kill < Soldiers(victim).effectiveness
170
171             deathFlag = 1;
172             afterLife(player,afterlife); % REGENERATION OR NOT
173
174         end
175
176         updateWorld();
177
178         if video
179
180             clf(h,'reset')
181             displayWorld
182             currFrame =getframe(h);
183             writeVideo(vidObj,currFrame);
184
185         end
186
187     end
188
189 end
190
191 end
```

```
1  function [w] = createWorld()
2
3  % CREATEWORLD   function that creates the first layer of the world matrix.
4  %               This layer indicates the content of each block on the map.
5  global param
6
7  gridSize        = param.gridSize;
8  nos             = param.nos;
9  noc             = param.noc;
10 nol             = param.nol;
11 allowLeaders    = param.allowLeaders;
12 world           = ones(gridSize*gridSize,1);
```

```
13
14
15 world(1:nos)          = 2; %   soldiers
16 world(nos+1:nos+noc)  = 3; %   civilians
17
18 if allowLeaders==1
19
20     world(nos+noc+1:nos+noc+nol) = 7; %   leaders
21
22 end
23
24 tmp = randperm(gridSize*gridSize);
25
26 world  = world(tmp);
27 world2 = zeros(gridSize,gridSize);
28
29 tmp =randperm(gridSize);
30
31 for i=1:gridSize
32     j = tmp(i);
33     world2(:,i) = world((j-1)*gridSize+1:j*gridSize)';
34 end
35
36 w = zeros(gridSize,gridSize,2);
37 w(:,:,1) = world2;
38
39 end
```

```
1  function [Civilians,Soldiers,Leaders] = populateWorld()
2
3  % POPULATE WORLD    Locates civilians and soldiers in first layer of world
4  %                   matrix and creates their structures. Furthermore, the
5  %                   function saves the identity of each structure in a
6  %                   second layer of the world matrix. With this method, it
7  %                   is very easy to find which civilian or soldier is
8  %                   situated in each block.
9  global world
10 global param
11
12 if param.allowLeaders == 0
13
14     Leaders = [];
15
16 end
17
18 worldLayer = world(:,:,1);
19
20 %   Find all points of the map, where civilians are supposed to be.
```

```
21 c = find(worldLayer == 3);
22
23 %   Create the civilians
24 for i=1:length(c)
25
26     [row col]    = ind2sub(size(worldLayer), c(i));
27
28     Civilians(i) = civilian(row,col);
29
30     world(row,col,2) = i;
31 end
32
33 %   Find all points of the map, where soldiers are supposed to be.
34 s = find(worldLayer == 2);
35
36 %   Create the soldiers.
37 for i=1:length(s)
38
39     [row col]   = ind2sub(size(worldLayer), s(i));
40     Soldiers(i) = soldier(row,col);
41
42     world(row,col,2) = i;
43
44 end
45
46 if param.allowLeaders==1
47
48     %   Find all points of the map, where leaders are supposed to be.
49     l = find(worldLayer == 7);
50
51     %   Create the leaders.
52     for i=1:length(l)
53
54         [row col]   = ind2sub(size(worldLayer), l(i));
55         Leaders(i) = leader(row,col);
56
57         world(row,col,2) = i;
58
59     end
60
61 end
62
63 end
```

```
1 function [] = afterLife(player,belief)
2
3 %   AFTERLIFE  function that performs the necessarry steps after a ...
      civilian's death
```

```
4  %
5  %   Input:  if belief == 0, There is no life after death
6  %           if belief == 1, we believe in reincarnation
7
8  global world
9  global param
10 global Civilians
11 global noc
12
13 gridSize  = param.gridSize;
14 tempWorld = world(:,:,2);
15
16 x = Civilians(player).x;
17 y = Civilians(player).y;
18
19 world(x,y,1) = 1;           % R.I.P
20 world(x,y,2) = 0;
21
22 if belief == 0
23
24     Civilians(player) = [];
25
26     noc = noc - 1;
27
28     tempWorld(tempWorld > player) = tempWorld(tempWorld > player)-1;
29     world(:,:,2) = tempWorld;
30
31 elseif belief == 1
32
33     while 1
34
35         pos       = randi(gridSize*gridSize);
36         [row col] = ind2sub(size(tempWorld), pos);
37
38         if tempWorld(row,col) == 0
39
40             break   % Exit the loop when the random place of ...
                   regeneration is empty
41
42         end
43
44     end
45
46     Civilians(player) = civilian(row,col);  % generate civilian with new ...
           characteristics somewhere else
47
48     world(row,col,2)  = player;
49
50     % Follows part of the function updateWorld but just for one civilian
51
```

```
    updateWorld(player)

end


end
```

```
function [] = updateCivilian(data,damage) % (index of civilian, how many ...
    people were injured)

%UPDATECIVILIAN     Summary of this function goes here
%                   Detailed explanation goes here

global Civilians

fearBaseModifierV  = 0.1;
fearBaseModifierS  = 0.1;
angerBaseModifierV = 0.03;
angerBaseModifierS = 0.03;

fearModifierRich    = 1.1;
fearModifierMiddle  = 1;
fearModifierPoor    = 0.9;

angerModifierRich       = 0.8;
angerModifierMiddle     = 1;
angerModifierPoor       = 1.2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
civ     = data(1);
injured = data(2);

fear  = Civilians(civ).fear;
anger = Civilians(civ).anger;

if strcmp(Civilians(civ).wealth,'Rich')
    fearIncrement   = fearModifierRich;
    angerIncrement  = angerModifierRich;
elseif strcmp(Civilians(civ).wealth,'Middle')
    fearIncrement   = fearModifierMiddle;
    angerIncrement  = angerModifierMiddle;
else
    fearIncrement   = fearModifierPoor;
    angerIncrement  = angerModifierPoor;
end

if injured == 1

```

```
41      Civilians(civ).fear  = fear+fearIncrement*fearBaseModifierV*(1-fear);
42      Civilians(civ).anger = ...
            anger+angerIncrement*angerBaseModifierV*damage*(1-anger);
43
44  elseif injured == 0
45
46      Civilians(civ).fear  = fear+fearIncrement*fearBaseModifierS*(1-fear);
47      Civilians(civ).anger = ...
            anger+angerIncrement*angerBaseModifierS*damage*(1-anger);
48
49  end
50
51  if Civilians(civ).fear > 1
52
53      Civilians(civ).fear = 1;
54
55  elseif Civilians(civ).fear < 0
56
57      Civilians(civ).fear = 0;
58
59  end
60
61  if Civilians(civ).anger > 1
62
63      Civilians(civ).anger = 1;
64
65  elseif Civilians(civ).anger < 0
66
67      Civilians(civ).anger = 0;
68
69  end
70
71  end
```

```
1   function [] = updateWorld(index)
2
3   %UPDATEWORLD    Updates maps values and agents colours (for displaying ...
        purposes)
4   %               according to the state of each civilian.
5   %               If input index is specified, only the corresponding ...
        civilians
6   %               will be updated.
7
8   global noc
9   global world
10  global Civilians
11
12  if nargin<1
```

```
13
14      for i = 1:noc
15
16          anger = Civilians(i).anger;
17          fear  = Civilians(i).fear;
18          thres = Civilians(i).thres;
19
20          if anger < fear
21
22              value = 3;      % Green
23              Civilians(i).Colour = [0.5 1 0.5];
24
25          elseif (anger ≥ fear) && (anger ≤ thres)
26
27              value = 4;      % Yellow
28              Civilians(i).Colour = [ 1 1 0];
29
30          elseif (anger ≥ fear) && (anger ≥ thres) && (Civilians(i).attack ...
                == 0)
31
32              value = 5;      % Orange (Latent insurgent)
33              Civilians(i).Colour = [ 1 0.5 0];
34
35          elseif (anger ≥ fear) && (anger ≥ thres) && (Civilians(i).attack ...
                == 1)
36
37              value = 6;      % Red (Active insurgent)
38              Civilians(i).Colour = [ 1 0 0];
39
40          end
41
42          world(Civilians(i).x,Civilians(i).y,1) = value;
43
44          if value == 5 || value == 6
45
46              Civilians(i).threat = 1;
47
48          end
49
50      end
51
52  else
53
54      for i = 1:length(index)
55
56          anger = Civilians(index(i)).anger;
57          fear  = Civilians(index(i)).fear;
58          thres = Civilians(index(i)).thres;
59
60          if anger < fear
```

```
61
62              value = 3;      % Green
63              Civilians(i).Colour = [0.5 1 0.5];
64
65          elseif (anger ≥ fear) && (anger ≤ thres)
66
67              value = 4;      % Yellow
68              Civilians(i).Colour = [ 1 1 0];
69
70          elseif (anger ≥ fear) && (anger ≥ thres) && ...
                (Civilians(index(i)).attack == 0)
71
72              value = 5;      % Orange (Latent insurgent)
73              Civilians(i).Colour = [ 1 0.5 0];
74
75          elseif (anger ≥ fear) && (anger ≥ thres) && ...
                (Civilians(index(i)).attack == 1)
76
77              value = 6;      % Red (Active insurgent)
78              Civilians(i).Colour = [1 0 0];
79
80          end
81
82          world(Civilians(index(i)).x,Civilians(index(i)).y,1) = value;
83
84          if value == 5 || value == 6
85
86              Civilians(index(i)).threat = 1;
87
88          end
89
90      end
91
92  end
93
94  end
```

```
1
2   function [agents] = findAgents(x,y,type,interactionGrid)
3
4   %FINDAGENTS     This function finds all the agents of type = 'type'
5   %               (civilians,soldiers) in a grid starting from (x,y) and
6   %               extending relevant to the interactionGrid.
7   %
8   %   Inputs      x,y : coordinates of interest
9   %               type: kind of agents we are looking for ...
        ('soldiers','civilians')
10  %               interactionGrid: area of interaction
```

```
11 %
12 %   Output      agents: vector containing the indices to the agents
13
14 global world
15 global param
16
17 gridSize = param.gridSize;
18
19 % Find the limits of the grid to be scanned
20
21 x1 = x - floor(interactionGrid/2);
22 x2 = x + floor(interactionGrid/2);
23 y1 = y - floor(interactionGrid/2);
24 y2 = y + floor(interactionGrid/2);
25
26 % Make corrections in case numbers are out of bounds
27
28 if x1<1
29     x1 = 1;
30 end
31 if x2>gridSize
32     x2 = gridSize;
33 end
34 if y1<1
35     y1 = 1;
36 end
37 if y2>gridSize
38     y2 = gridSize;
39 end
40
41 % Find agents
42 agents = [];
43
44 for i=x1:x2
45     for j=y1:y2
46
47         if world(i,j,1)==2 && strcmp(type,'soldier')
48
49             agents(end+1) = world(i,j,2);
50
51         elseif world(i,j,1) > 2 && world(i,j,1)<7 && strcmp(type,'civilian')
52
53             agents(end+1) = world(i,j,2);
54
55         end
56     end
57 end
58
59 end
```

```
function  moveInsurgent(civ)

%   moveInsurgent   This function attempts to find a new position for a ...
    latent or
%                   active insurgent, when it not already possible to ...
    make any
%                   attacks.

global world
global param
global Civilians

AttackGrid = param.AttackGrid;
gridSize   = param.gridSize;

tempWorld  = world(:,:,2);
foundPlace = 0;

while 1

    %   pick a random position on the map
    pos       = randi(gridSize*gridSize);
    [row col] = ind2sub(size(tempWorld), pos);

    %   check if is empty and there is a soldier close by
    if tempWorld(row,col) == 0 && ¬...
        isempty(findAgents(row,col,'soldier',AttackGrid))

        foundPlace = 1;
        break   % Exit the loop when the random place of regeneration is ...
            empty

    end

end


if foundPlace == 1

    x = Civilians(civ).x;
    y = Civilians(civ).y;
    %     fprintf('Insurgent at position (%d,%d) moves to ...
        (%d,%d).\n',x,y,row,col)
    world(row,col,1) = world(x,y,1);
    world(row,col,2) = civ;
    world(x,y,1) = 1;
    world(x,y,2) = 0;
    Civilians(civ).x = row;
    Civilians(civ).y = col;
```

```
45
46 end
47 end
```

```
1  function outOfRangeFlag = outOfRange
2
3  %outOfRange Returns true if no insurgent can perform an attack on a soldier
4  %           or false otherwise.
5
6  global param
7  global Civilians
8
9  AttackGrid = param.AttackGrid;
10 outOfRangeFlag = true;
11
12 for i=1:length(Civilians)
13
14     if Civilians(i).threat == 1
15
16         nearbySoldiers = ...
               findAgents(Civilians(i).x,Civilians(i).y,'soldier',AttackGrid);
17
18         if ¬isempty(nearbySoldiers)
19             outOfRangeFlag = false;
20             break
21
22         end
23     end
24
25 end
26
27 end
```

```
1  function propaganda( civ,influence )
2  %propaganda This function implements the propaganda action of a leader.
3  %           Inputs :
4  %                   - civ:  a vector with the indices of civilians in the
5  %                           influence range of the leader.
6  %                   - influence: how much the leader affects the civilians.
7  global Civilians
8
9  for i=1:length(civ(i))
10
11     Civilians(civ(i)).anger = Civilians(civ(i)).anger + ...
           influence*(1-Civilians(civ(i)).anger);
12     Civilians(civ(i)).fear  = Civilians(civ(i)).fear  - ...
           influence*(1-Civilians(civ(i)).fear);
```

```
13
14      if Civilians(civ(i)).fear > 1
15
16          Civilians(civ(i)).fear = 1;
17
18      elseif Civilians(civ(i)).fear < 0
19
20          Civilians(civ(i)).fear = 0;
21
22      end
23
24      if Civilians(civ(i)).anger > 1
25
26          Civilians(civ(i)).anger = 1;
27
28      elseif Civilians(civ(i)).anger < 0
29
30          Civilians(civ(i)).anger = 0;
31
32      end
33
34  end
35
36
37  end
```

```
1   function [Civilians] = recruit(Civilians)
2
3   %   RECRUIT : This function recruits a civilian among those who are angry
4   %   but not angry enough, and converts him into a latent insurgent.
5
6   global world
7
8
9   tempWorld  = world(:,:,2);
10
11  candidates = find(tempWorld == 4);                % find civilians with ...
        anger>fear but not latent
12
13  choose = randi([1 length(candidates)],1,'uint32');% pick a random one
14
15  chosen = candidates(choose);                      % pick a random one
16
17  [x y] = ind2sub(size(tempWorld),chosen);
18
19  chosen = tempWorld(x,y);
20
21  Civilians(chosen).anger = Civilians(chosen).thres + 10*eps ;
```

```
22
23 if Civilians(chosen).anger > 1
24
25     Civilians(chosen).anger = 1;
26
27 end
28
29 updateWorld(chosen);
30
31 end
```

```
1  function displayWorld(ind,inj)
2
3  %   displayWorld    This function will simply display the current state of
4  %                   the world.
5  global param
6  global noc
7  global Civilians Soldiers Leaders
8
9  nos             = param.nos;
10 nol             = param.nol;
11 gridSize        = param.gridSize;
12 CollateralGrid  = param.CollateralGrid;
13
14 for i=1:nos
15
16     plot(Soldiers(i).x,Soldiers(i).y, ...
17         'Marker','.', 'MarkerSize',15, ...
18         'Color',Soldiers(i).Colour);
19     hold on
20
21 end
22
23 for i=1:noc
24
25     plot(Civilians(i).x,Civilians(i).y, ...
26         'Marker','.', 'MarkerSize',15, ...
27         'Color',Civilians(i).Colour);
28     hold on
29
30 end
31
32 for i=1:nol
33
34     plot(Leaders(i).x,Leaders(i).y, ...
35         'Marker','.', 'MarkerSize',15, ...
36         'Color',Leaders(i).Colour);
37     hold on
```

```
38
39 end
40
41 if nargin≥1
42
43     plot(Civilians(ind(1)).x,Civilians(ind(1)).y, ...
44         'Marker','s', 'MarkerSize',15, ...
45         'Color',[1 0 0]);
46     hold on
47
48     plot(Soldiers(ind(2)).x,Soldiers(ind(2)).y, ...
49         'Marker','s', 'MarkerSize',15, ...
50         'Color',[0 0 0]);
51     hold on
52
53     for i=1:length(inj)
54         if inj(2,i)==1
55             plot(Civilians(inj(1,i)).x,Civilians(inj(1,i)).y, ...
56                 'Marker','d', 'MarkerSize',15, ...
57                 'Color',[1 0 1]);
58             hold on
59         end
60     end
61     rectangle('Position',[Civilians(ind(1)).x-floor(CollateralGrid/2),Civilians(ind(1)).y-floo
62 end
63 axis([-1 gridSize+1 -1 gridSize+1 ]);
```

```
1  % function f1 = ...
       displayResults(active,latent,civilians,deaths,attacks,recruits,timeSteps)
2  function f1 = displayResults(Results)
3  %   displayResults:
4  %               This function is used at the end of a simulation to display
5  %               the main results.
6  global param
7  global noc
8  global Soldiers
9  global Civilians
10 global Leaders
11
12 nos = param.nos;
13 nol = param.nol;
14 allowLeaders = param.allowLeaders;
15 gridSize = param.gridSize;
16 CollateralGrid = param.CollateralGrid;
17 AttackGrid = param.AttackGrid;
18 InfluenceGrid = param.InfluenceGrid;
19 afterlife = param.afterlife;
20
```

```
ASPECT_RATIO = 1680/850;

if (exist('CURRENTFIGURE','var')==0)
    % get screen resolution
    scr=get(0,'ScreenSize');
    % and force a 16:9 display port, so the plots will look the same on
    % all computer systems and screens. optimized for: win seven
    % X/Y of bottomleft-corner, width, height of figure
    if (floor(scr(3)/ASPECT_RATIO)>scr(4)-160)
        FIGUREPOS = [10; 60; floor((scr(4)-160)*ASPECT_RATIO); scr(4)-180];
    else
        FIGUREPOS = [10; 60; scr(3)-20; floor((scr(3)-20)/ASPECT_RATIO) ];
    end
    CURRENTFIGURE = 1;
    % converts action-radius to pixel
    GRAPHSCALING = FIGUREPOS(3)/1680 * 20;
    clear scr;
end

f1 = figure('position', FIGUREPOS, ...
    'CurrentObject',CURRENTFIGURE, ...
    'Name','Graphical Output of Insurgency Evolution', ...
    'NumberTitle', 'off', ... % removes "figure 1:"
    'Color',[1 1 1]); % background color: white


% get the positions of the world map, depending on world size
if (exist('worldplot','var')==0)

    maxwidth = 0.42;
    maxheight = maxwidth*ASPECT_RATIO;
    width = maxwidth;
    height = maxheight;
    worldplot = [0.46-width 0.925-height width height];
    clear width height maxwidth maxheight;

end

axes1 = axes('Parent',f1, 'Position',worldplot, ...
    'Color',[1 1 1]);
box(axes1,'on'); % draw a black border around the plot
hold(axes1,'all');
axis(axes1, [-1 gridSize+1 -1 gridSize+1]);

for i=1:nos

    plot(Soldiers(i).x,Soldiers(i).y, ...
        'Marker','.', 'MarkerSize',15, ...
        'Color',Soldiers(i).Colour);
    hold on
```

```
71
72  end
73
74  for i=1:noc
75
76      plot(Civilians(i).x,Civilians(i).y, ...
77          'Marker','.', 'MarkerSize',15, ...
78          'Color',Civilians(i).Colour);
79      hold on
80
81  end
82
83  for i=1:nol
84
85      plot(Leaders(i).x,Leaders(i).y, ...
86          'Marker','.', 'MarkerSize',15, ...
87          'Color',Leaders(i).Colour);
88      hold on
89
90  end
91
92  title({'Agents World'},'FontWeight','bold');
93
94  %% Plot Graphs
95  active    = Results.active;
96  timeSteps = Results.timeSteps;
97  latent    = Results.latent;
98  attacks   = Results.attacks;
99  deaths    = Results.deaths;
100 civilians = Results.civilians;
101 recruits  = Results.recruits;
102
103 Evolution = [   worldplot(1)+worldplot(3)+0.08, ...
104     worldplot(2)+worldplot(4)-0.4, ...
105     1-worldplot(1)-worldplot(3)-0.1 , 0.4];
106
107 axes2 = axes(   'Parent',f1, 'Position',Evolution, ...
108     'Color',[0.94 0.94 0.94]);
109
110 box(axes2,'on');
111
112 plot(1:timeSteps,active(1:timeSteps),'r',1:timeSteps,latent(1:timeSteps),'c',...
113     1:timeSteps,attacks(1:timeSteps),'b',1:timeSteps,deaths(1:timeSteps),'m',...
114     1:timeSteps,civilians(1:timeSteps),'y','LineWidth',3);
115 legend('# of active insurgents','# of latent insurgents','# of ...
        attacks','# of deaths','# of civilians')
116 hold on
117 if allowLeaders == 1
118     plot(1:timeSteps,recruits(1:timeSteps)','g');
```

```
    legend('# of active insurgents','# of latent insurgents','# of ...
        attacks','# of deaths','# of civilians','# of recruitments')
end
xlabel('Time Steps')
title('Evolution of Insurgency','FontWeight','bold')

%% Display Simulation Parameters
par=[ Evolution(1) worldplot(2) 0.4 Evolution(2)-worldplot(2)-0.15];
% par=[ Evolution(1) worldplot(2) 0.4 Evolution(2)-worldplot(2)-0.01];
% Create textbox
annotation(f1,'textbox',...
    [par(1)-(Evolution(1)-(worldplot(1)+worldplot(3))) par(4) par(3) ...
        par(4)-0.07],...
    'String',{'Important Parameters'},...
    'FontWeight','bold',...
    'HorizontalAlignment','center',...
    'LineStyle','none');

annotation(f1,'textbox',...
    [par(1)+0.05 par(2)+0.14 par(3)-0.25 par(4)-0.05],...
    'String',{'Civilians Start:','Civilians End:','World Size:',...
    'Attack Radius:','Collateral Radius:','Influence Radius:','Time ...
        Steps:','Replacement:',...
    'Allow Leaders:','Leaders:'},...
    'FitBoxToText','off');

annotation(f1,'textbox',...
    [par(1)+0.12 par(2)+0.01 par(3) par(4)+0.08],...
    'String',{num2str(500),num2str(noc),...
    [num2str(gridSize) 'x' num2str(gridSize)],...
    num2str(AttackGrid),...
    num2str(CollateralGrid),...
    num2str(InfluenceGrid),...
    num2str(timeSteps),...
    num2str(afterlife),...
    num2str(allowLeaders),...
    num2str(nol)},...
    'FitBoxToText','on',...
    'EdgeColor','none');


end
```

```
function [x] = gsample(mu,Sigma,n)

%GSAMPLE  Function that produces n draws from a D-dimensional
%         Gaussian. (The dimension D is implicitly specified
%         by mu and Sigma)
```

```
6
7  D = length(mu);
8
9  [U,L] = schur(Sigma); % Sigma = U*L*U'
10
11 g = randn(n,D); % n draws, D dimensions N(0,1)
12
13 x = U*sqrt(L)*g'+repmat(mu,n,1)';
14 x = x';
15
16 end
```