Position Papers of the Federated Conference on
Computer Science and Information Systems pp. 249–256

DOI: 10.15439/2015F417
ACSIS, Vol. 6

Multi-agent simulation of the world found in the G. R. R. Martin's novel "Sandkings"

Jakub Ciecierski, Viet Ba Mai, Michał Słupczyński and Wojciech Zyskowski
Faculty of Mathematics and Information Science, Warsaw University of Technology
Plac Politechniki 1, 00-660 Warsaw, Poland

***Abstract*—George R. R. Martin's novel entitled *Sandkings* introduces tribes of ant-like creatures which, when locked in a terrarium deprived of food, fight for survival. The tribes differ in the color of their armor. However, in the novel, despite the extreme conditions, they've never attempted to kill the queens of different fractions for *cannibalistic* purposes. Approaching the situation from the perspective of profit-driven logic, lack of such behaviour is highly questionable. In multiple cases, attacking another Maw to eat her could improve the odds of survival in a hostile environment.**

To provide an initial attempt to answer the question of whether or not the queens should order an attack and try to "eat each other", we have developed a multi-agent simulation based on the novel. Through analysis of its results we hoped to prove hypothesis that with the increased hostility of environment the *Sandkings* would develop cannibalistic behaviours and fight until eventually only one tribe is left.

I. Introduction

SAND KINGS is a science-fiction novel written by George R.R. Martin in 1979 [1]. There, Simon Kress, the main protagonist of the story, is as a collector of lethally dangerous, exotic, and mostly extraterrestrial, animals. Due to his prolonged business trips, said animals often die in the span of his absences. Eventually, when the need for another replacement occurred, Kress stumbled upon a terrarium filled with, what the shopkeeper described as, four colonies of *Sandkings*.

Each colony consisted of an immobile female, queen Maw, and a number of ant-like Mobiles, which are controlled by their Maw through telepathy. In order to survive, the tribe mothers need to be fed regularly, thus one of the Mobile's main purposes is to hunt down and collect food for her to digest. Luckily, the Maws are able to eat "anything" (other than sand), so everything that can be found in the terrarium can be brought to her. On the other hand, the Mobiles are not able to feed on their own, they can be only fed by the Maw, hence the second priority for mobiles is Maw defense, as when their mother dies, they will also perish. Throughout this paper we will use terms Sandkings and Mobiles interchangeably.

The shopkeeper informed Kress that, over time, the four colonies would start to wage wars between each other. Excited with this vision, Simon bought all four Sandking colonies and decided to have them installed in the living room of his flat. With time, the new owner hosted parties to show off his new pupils, and he couldn't wait for the conflicts to emerge. As Sandkings lived peacefully for the days to come, he started to starve them so they would become desperate.

From graceful and highly intelligent entities, the Sandkings turned into wild and murderous creatures that sought only to find more food to grow. Eventually, due to an unlucky event, they have broken out of the terrarium they were imprisoned in, and proved to be a threat not only to animals thrown into the terrarium, but also to their owner and to other people.

In our opinion, a story like the one which has been depicted above is not plausible, especially in the phase just before the Sandkings got free. However, being strangely attracted to the novel, we have decided to model the stage of the story, that took place slightly after Sandkings were in an extreme starvation period. This means that the only goal, which was driving the Sandkings, was to collect as much food as possible. In order to do so, the Mobiles not only would try to kill any living animals that were thrown into the terrarium, but would also fight with each other and potentially kill and eat other Maws – if that would prove profitable.

The needed model (and simulation) could most naturally be done using software agents. The application simulating the, defined above, situation could easily be used to predict plausible Sandkings behaviour(s). The main goal of our investigation was to resolve any doubts regarding lack of cannibalistic actions taken by the Maws towards "other tribes".

We proceed as follows. First, we shortly describe related work and tools used in our simulation. We follow by a description of the structure of the program and entities used in our simulation. Next, in Section IV, we list technological solutions used to develop the model. Following Section V, describes two scenarios and datasets used in them, (i) the friendly environment data set, and (ii) the deadly environment dataset. We conclude this section with overview of experimental results. Finally, we summarise our results present the resolution of our hypothesis.

A. State of the art

In nature, ants are social insects that, as individuals, are not capable of performing complex task(s), as they are "bounded" by their limited memory and behaviour that seems to have a noticeable random component. However, when the collective behaviour of ants is considered, maintaining pheromone-based communication, they prove capable of performing complicated tasks, like colony protection, or transporting food too heavy for an individual. This shows that even with very limited amount of computational resources per "agent" (ant), a swarm of ants can efficiently solve advanced tasks. Because of this,

researchers often make use of ant-like agent modelling, most often for optimisation purposes.

For instance, ant-like agents, used for load balancing in telecommunications networks, have been proposed by Ruud Schoonderwoerd, Owen Holland and Janet Bruten. If there is too much traffic through some network's node it might lead to loss of calls. Agents, whose behaviour has been modelled based on ants communicating by means of pheromones, traverse the network picking their path accordingly to pheromone distribution at each node, leaving an appropriate trail with each move. Such ant based model proved to be more efficient than shortest-path algorithms, or algorithm-based mobile agents [6].

Other use of ant-like agents is proposed by Stephen C. Pratt, David J. T. Sumpter, Eamonn B. Mallon and Nigel R. Franks in "An agent-based model of collective nest choice by the ant Temnothorax albipennis". The ants mentioned there are known for their ability to collectively choose the best nest site, out of several available, even if most of the ants have not visited more than one location. After finding a new site, the finding it ant tries to assess quality of the nest and convince the other active ants to move the colony. When new ants visit a location, adding to its temporary population, they reassess it by again trying to convince new ants. This might prove to be a kind of collective decision, where the best potential sites have the biggest temporary population. With proper empirical data, the algorithmic form of a collective decision-making mechanism can be captured and easily modelled [7].

Finally, the Biomass tool [8] allows to design ecosystem experiments. This agent based model simulation focuses on exploring the relationships between population in a given ecosystem. The individual decisions are based on environmental conditions. Finally the tool provides a way to configure the population by parametrization without having to program it manually. This work is the closest to our approach.

II. Tools

The *Repast Simphony* [3] is an open source agent-based modeling toolkit that simplifies model creation and experimentation. Out of a variety of accessible tools, we have chosen Repast for the development of the simulation. This was mainly due to its simplicity of use and the possibility of run-time dynamic interactions with the simulation. A big additional bonus for the choice of this modeling toolkit was the continued development and support of this package, which when compared to other agent-based modeling and simulation toolkits is a rare commodity.

Even though the *Repast Toolkit* supports many programming languages, including, among others, C#, VB.NET, Python, C++, Prolog, we decided to implement the application in the Java Runtime Environment. The most relevant arguments behind this decision include multi-operating system support, which Java provides and the fact that other programming languages were discouraged [5]. Apart from this, the developers of the *Repast Toolkit* suggest the usage of the *Eclipse* IDE, and thus we have followed their advice.

The creation process of 2D Euclidan environments and agents, with specific graphical properties, is particularly easy in the *Repast Simphony*. Specifically, an entity extending the class Agent is already ready for further development. Hence, the developer does not have to worry about implementation of the agent itself, there is only need to focus on general architecture of the agent based simulation.

In addition to this, another useful feature is a fully concurrent, discrete event scheduler, which provides a straightforward method of determining each agent type's behaviour on every simulation step.

Tutorial materials [2] provided by the Repast Team, especially the implementation of a zombie epidemic simulation, were the initial foothold in the topic of multi-agent based simulation. The extent and quality of document commentary significantly sped-up the process of simulation development.

III. Program structure

Agent-based computer systems are inherently object oriented, which allows for modularity and ease of development. In order for the simulation setup to closely resemble the universe created by *George Martin*, the modeling application has been divided into the following entities: Maw, Mobile, Formation, Food and Enemy.

A. Agents

Let us start with the description of the agent-based model, often referred to as ABM. Formally, it is a computational method that enables to create, analyse, and experiment with models composed of agents that interact within an environment. Most typically, they are used in social sciences, where researchers try to depict simplified versions of phenomena that are looked into. Based on a specific set of inputs, the model calculates resulting output data, which often are used to confirm or decline some theory [6], [8]. The implemented ABM is a system composed of multiple agents, which sustain interaction and communication. Such construct is called Multi-Agent system, otherwise referred to as a MAS. The agents relevant to our work are as follows.

1) Maw: The *Maw* (referred to as Mother or Queen) is an immobile agent, which spawns the Mobiles, described in the next section. Maw's life depends on the Mobiles ability to feed it, as once a Maw dies, so do all her Mobiles.

Four Mothers exist in our simulation (as shown in Figure 1: denoted as 1a, 1b, 1c and 1d). Every one of them, as it is supposed to exhibit a Hive-Mind behaviour, is responsible for coordinating the actions of her Mobiles. The children of each Maw inform her about every object found in the environment, possible threats and available food, allowing her to collect an extensive knowledge database. The types of threats and food are described in the Section III-B.

Based on the available information, each Maw organises the work of her children. This is done by means of a list of tasks (or assignments), which is sorted by the most profitable and least dangerous assignments being placed first. Such tasks have the purpose of keeping her fed, safe and alive. Possible

tasks are not limited to picking up heavy food, which single Mobiles are not able to move, or killing off monsters that can be later eaten, but also attacking other Maws in hopes for the acquisition of their food.

As picking up food is not related to any specific danger, except any possible threats met on the way, it is always considered the least dangerous and thus has the best profit potential. Attacking and killing an enemy is usually considered mediocre dangerous, where the profit is equal to amount of food that can be collected from the enemy and the danger is relative to the strength of the enemy. Finally, the profit of killing a Maw is considered. This involves the amount of food that can be gained by killing each of her children and the food that she actually has, while the danger is related to the overall strength of all her mobiles.

Depending on the strength, which is required to finish the task, the Maw might try to create a temporary alliance with another tribe. The lifespan of this alliance usually does not exceed the task greatly, as the profitability of another alliance may be calculated at any later stage of the simulation.

When one of the Maw's population grows bigger than the rest combined, it becomes a threat to other Maws. This can lead to an alliance constructed between the weaker Maws in order to start a war against the stronger fraction. Such behaviour is supposed to keep the strength of all Maws in balance.

2) Mobiles: The second most important element of the simulation are the Mobiles, as they act as a set of actuators for their Maw.

With simple autonomy, they prioritize the completion of the task they were assigned over their own survival, Mobiles wander around the terrarium looking for "points of interest". Every piece of food which is light enough to be picked up by a single Mobile is picked up and carried back to the fraction's Maw. Food, which is too heavy, and Enemies which are too strong to fight, are added to the Knowledge Base and are scheduled to be told to the Maw on the next occasion (i.e. when the Mobile goes home next time).

If an Enemy (in our simulation this is, most likely, a dangerous animal – see III-A3) is spotted, Mobiles will charge towards it, provided it is weak enough, and try to kill it in order to bring its leftovers to their Maw as food (see III-B2), or flee if fight would prove to be too dangerous.

As Mobiles are unable to digest on their own, they have to bring the food firstly to the Maw and only then she can feed them. Because of this fact, unless they have task to complete on the other side of terrarium, they tend to stay nearby their mother, keeping her safe and themselves fed.

3) Enemy : The opponents of Maws and Mobiles are the Enemies, also referred to as Monsters or Creatures. They are agents of much higher strength and health than a Mobile. They move on the Map in random directions, but do not eat anything they find, because their only purpose in the simulation is to fight with the Mobiles.

As shown in Figure 1, the following three types of Enemies exist in the simulation: *Spiders* [3c], *Scorpions* [3b], and the strongest ones – *Snakes* [3a]. As the simulation proceeds, the "damage level" of Enemies increases, depending on the current tick count of the simulation. The visual size of the enemies reflects their damage level (the more damaged they are, the smaller they get). This also indicates the number of Mobiles needed to kill them.

4) God: This is an "invisible agent" which has no graphical representation in the simulation. In each step, *God* may drop either Food, or an Enemy, in the terrarium, at a random point on the grid. The God "appears" every 100th tick of a simulation. Specific types of Food and Enemies to be spawned are chosen at random, where the probability of dropping them depends directly on their properties – the more powerful they are / the more food-value they have, the smaller the chance of their appearance. For example, *Pizza*, which is the most beneficial Food, for the Mobiles, and *Snake*, which is the strongest creature in the simulation have the least likelihood of them being spawned. Detailed properties of each entity, used in actual simulations, can be found in Section V.

The God module is also responsible for increasing the damage level of the enemies, as described in more detail in the Section III-A3. Additionally, a constraint was introduced such that the Monsters may never be created nearby the Maws, to prevent mobiles from dying immediately.

As opposed to the *Mobiles* and the *Maws*, neither *God* nor an *Enemy* have an ability to learn about the environment.

B. Environment

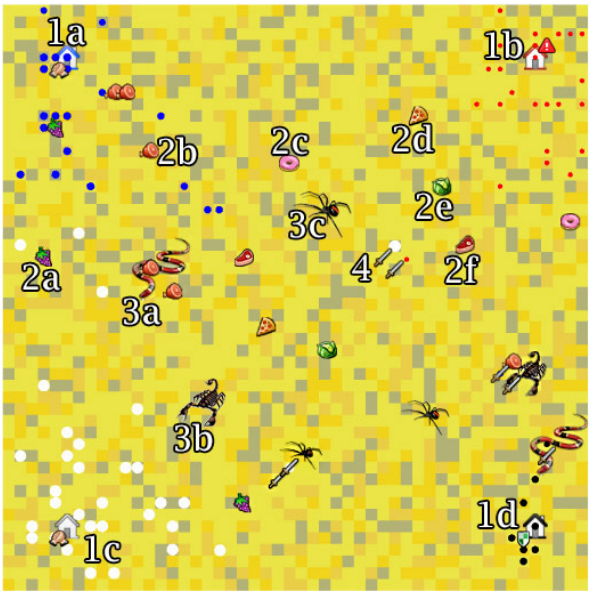

Fig. 1. Simulation with all elements.

The Environment of our simulation has been developed to represent the terrarium, from the *Sandkings* novel, as much as possible. As the simulation starts a Map representing sand and rocks is created, as well as Maws, which spawn their Mobiles during the runtime.

1) Map: The scenery of our simulation is a grid consisting of cells painted with different shades of yellow and grey, to graphically represent a simplified version of the terrarium, which was filled with sand and rocks. To make observation easier there exist icons representing an event shown for a given timeout (usually a few ticks – 50, 100, ... to grab attention but not to disrupt the visual flow of the simulation).

As seen in the Figure 1, the following icons – *Shield*[1d], *Shaking hands*[1a, 1c] and *Warning sign*[1b] concern Maws, hence they will appear right next to them. These icons indicate, respectively, that the Mother needs to be defended, has made an Alliance with another race, or that she is starving. The shield shows up when a Maw is in danger, which means that an opponent, either an Enemy or another fraction is about to attack her. In the case of the fourth sign, the *Sword*(Figure 1, [4]) will pop up next to every Mobile that is in a fight. The last informative picture is a *Grave stone* which appears on the Map in place of a dead Maw. Differently from the other icons, it does not have a timeout. At the end of a simulation, when only one Maw is left – as the others have died by hunger or during a war, an ending screen appears in the middle of a Map. It also contains information about which Maw was the last to stay alive.

2) Food : Food is an object (on the Map), which does not move. In our simulation, six types of Food with different, yet proportional, weight and "calorie count" properties are introduced. Weight property of each Food category affect how many Mobiles are needed to carry that piece of Food, while calories represent how much the "Maw's food repository" is increased by eating it. Using eaten food the Maw may either give birth to a new mobile or increase the overall tribe strength, making each Mobile stronger. However, note that the overall Food needed to feed the Mobiles may be larger thus introducing risk of starvation.

The first four types of Food are dropped by the God agent. As shown in Figure 1 they are represented with icons of *Grape*[2a], *Doughnut*[2c], *Pizza* [2d] and *Cabbage* [2e] .

The last two types of Food are placed in the grid only (and every time) when either a Maw, Mobile or Enemy dies. When a Mobile dies, it turns into one piece of Food, shown as *Meat* (Figure 1, [2b]). For balancing purposes, a piece of Meat has less calorie value than any other kind of Food, and amounts to roughly half of the upkeep needed to feed one Mobile. Both Maws and Enemies become *Steaks* (Figure 1, [2f]) after their death. This means that the strength of an agent is directly proportional to the Food this agent drops when dying.

Additionally, dead Maws also drop extra Steaks proportionally to the amount food she has eaten throughout the entire simulation. The detailed values of each Food's properties, used in actual simulations, has been specified in Section V.

C. Formations

In order to model agent cooperation, a simple Mobile formation logic has been implemented. Specifically, a temporary agent called Formation is spawned for the duration of a specific Task. Every Mobile, which is in a Formation, is not allowed to carry out any of the movement, item carrying or fighting logic associated with a single Mobile. Instead, the Formation object is responsible for the simulation algorithms of all Mobiles in its ranks.

A new instance of a Formation gets instantiated when one of the Maws has a Task, in her Scheduler, which is too "difficult" for a single Mobile, i.e. *object too heavy to carry* or *enemy too dangerous to attack*. This allows for easy planning, basic tactics and task assignment. Formations are assembled by the respective Maws and are provided with all relevant information (during assembly of the Formation): the Task type (fetch food, attack an enemy), the size needed for the Formation to be formed and the Task location (the last known position of the food or enemy). To find the number of Mobiles necessary for the creation of a Formation, the Maw looks for any of her children, which are currently not assigned to any other Task, and are in her vicinity.

If the Task needs more Mobiles than available to the Maw (at a given moment), the Alliance system comes into play – Mothers of different fractions are asked whether they want to join the Task. If the asked Maw deems this profitable, she creates a Formation of her own. When the critical number of Mobiles for the given Task is reached, the Formation assembly process is finished and the Formation is sent out to carry out its Task. All Formation in a multi fraction Alliance have their movement synchronised so that they will arrive at their destination at the same time.

A Formation is created "on the position of the Maw", which it belongs to. This is due to the fact that she has to have the possibility to oversee the Formation assembly process by assigning free Mobiles from its vicinity to the Task. The Formation waits until the Maw finds enough Mobiles from either her own ranks or through Alliances and all assigned Mobiles arrive at the location of the Formation.

A Formation, which has not yet arrived at its goal location, moves all its assigned Mobiles towards the goal location (in this case, the food or enemy, which was supposed to be picked up or attacked). If the Formation's goal is attacking something, all profitable Enemies in the vicinity are attacked in addition to moving to the set goal position. Similarly to the logic of the Mobiles, a Formation can have specific functions called on arrival at a specific location. For example, this could check the Food in the vicinity, pick it up and start heading back for home. In another example, all Mobiles could attack an Enemy, provided one is nearby.

IV. Methodology

In order for an agent to act with a logic driven approach, a concrete set of "tools/methods" has to be introduced. Here, tools/methods, used in our simulation, are described.

A. Fighting

Relations between each tribe in terrarium can be described as Hostile, Neutral or Friendly, with tribes dividing into each Maws faction and Enemy tribe. Initially, the relation between each of mentioned tribes is set to Neutral.

When a mobile stumbles upon other Mobile, Maw or Enemy, if the relation between them is not friendly, a fight between them will start. If an agent is fighting, instead of moving, during the tick it inflicts damage equal to its attack parameter to one nearby non-friendly agents, i.e. reduces its amount of health by the value of attack, prioritizing hostile agents. When there are no non-friendly agents left in the vicinity, fight ends with a victory. If agents nearby are too dangerous to fight, it flees. In case of victory, the Mobile restarts its previous task, in case of fleeing it will move in the other direction than the danger is in, as to avoid it.

When mobiles are grouped into a Formation, as described in Section III-C, they are not executing their individual tick steps. Instead, the Formation is managing their actions, acting accordingly to its own tick step(s). Similarly to mobile fighting, the fight starts if Formation encounters non-friendly agents nearby and might end either in victory if there are no non-friendly agents left, or a retreat decision. Formation decides to retreat from fight if there are less mobiles left than the half of initial Formation size. During the fight instead of moving during its tick step the Formation orders each subordinate mobile to attack, as it was itself in a fight.

B. Message Exchange

We have constructed a simple mechanism to allow for an inter-agent communication. This system does not use the communication functionality provided by the *Repast Simphony*, as we had to create a "non-standard communication mechanisms" (to match the need of our simulation).

Agents can send messages to each other. This is done by adding an abstract message object (i.e. packets – which have sender and recipient) to a global message queue. This queue, on notification, will process its contents, and fire specific handlers for each message type. Based on the sender's needs and the recipient's current state, a response is sent (or not).

In our simulation, Mobiles and Maws communicate between each other on regular basis – as often as the need for communication arises (for example, when a Mobile has found knowledge or is hungry). Since each Maw is an immobile agent, all knowledge about the environment comes from its children. In order for a Mobile to send any message, it has to be in a small distance from its potential recipient. Thus, if a Mobile wants to message its Maw, it has to "return back home". Upon return, a Mobile can send a specific message to its Maw. Sending an *Inform* message will simply cause the Maw to add a given piece of information to its Knowledge Base. Mobile can also send an *Ask For Food* message and wait for a response. In this case Maw can respond with either *Acceptance* or *Rejection* based on its food supplies.

Being true to the book, Maws have the ability to send messages without any distance restrictions (through telepathy). A given Maw can also start communicating with other Maws, if it is faced with a problem it can not solve alone, e.g. dealing with a strong Enemy creature. In such case, a series of *Ask For Alliance* messages is exchanged between all Maws in order to begin an Alliance and to send out Formations, which could potentially defeat this Enemy.

C. Knowledge Base

Both Mobiles and Maws gather information about their environment. A single piece of information encapsulates the following properties: the object of interest, its location and the time when it was discovered. This information is saved in, unique for every Agent, Knowledge Base.

Agents do not duplicate information in their knowledge bases. Upon encountering an object of interest, the Mobile checks whether it already knows about it. Each piece of information in the knowledge base can be marked as *useless*. Useless information is the one that has already been processed and no further action should ever occur based on it. This is explained in detail in section IV-D.

Mobiles explore the Map and learn about their surroundings. Each time a Mobile encounters something interesting, an information about this object is added to its knowledge base. Each Mobile can learn about other hostile Mobiles, Enemy creatures or Food scattered around the Map. Once they find something, they run back to their Maw to inform her about their findings. Both Maw and her Mobiles have their own Knowledge Bases.

As mentioned previously, Maws can not explore on their own. Thus all information comes from the Mobiles exploring the Map. The Knowledge Base is used to control the actions of the Maw's children; see, Section IV-D.

D. Scheduler

The Agents' behaviour is scheduled dynamically during the simulation runtime. To achieve this, each Agent is given its own Scheduler, responsible for assigning Tasks, based on its Knowledge Base. It runs in parallel with their Knowledge Base by going through *non-useless* information and determining the next course of action.

A Task is a sequence of steps, which an agent should follow in order to complete a task. Tasks are split into stages. A stage is built of an execution part, which determines the necessary steps of the agent, and the condition, which indicates the end of this stage, thus indicating the beginning of another one. When the task reaches its final stage, it finishes successfully, which in turn marks the information that initiated this task as *useless*. A Task can also be finished successfully in abnormal cases, when a given stage was impossible to execute. Abnormality occurs, for instance, when an Agent reaches the destination and the object of interest is no longer there. It could mean that similar Task has been completed by different Agent. However, the current task can be replaced with another one having higher priority. In this case, the current Task is stopped, but the information is kept in the Knowledge Base without modification. This allows to restart the Task when the Scheduler, in synchronisation with the Knowledge Base, has determined that the Task should, nevertheless, be completed.

For example, a Mobile can be assigned a Task to pick up Food from a location, which is stored in its Knowledge Base.

This task is split into the following stages: move to Food's location, pick up the Food, and return home. When the Mobile returns home safely, the task is finished successfully. Since Mobiles do not share any global knowledge, other Mobiles of the same fraction could already haven returned this Food. Of course, the same applies to Mobiles between different fractions. Hence it can happen, that the indicated location contains no Food at all. In this case the $pickupfood$ stage fails and this task is marked as finished successfully. Even though the Mobile did not deliver the food we do not want the Mobile to repeat the process of delivery when there is no food to be picked up.

A Mobile can also be assigned a Task to inform its Maw about given object. This can occur when the found Food is too heavy for it to pick it up alone, or when it sees an Enemy Creature or Formation. During execution of this Task, a Mobile will try to avoid combat, if only possible. Of course, it may happen that the returning Mobile encounters an Enemy and gets killed by it. In such case the information it "carried" is forgotten and has to be rediscovered by other Mobiles.

After being informed about the environment, Maw can generate a Task, which will control Mobile units. Execution of tasks for picking up heavy Food and fighting Enemy creature(s) has similar structure. In the first stage, Maw has to determine if given objective can be completed. In other words, if it has enough Mobiles to pick up food or to defeat certain Enemy. Then a Formation is constructed, which starts moving towards its objective. In case of Food, the formation is ordered to pick it up and return home. When the objective is to defeat an Enemy, the formation has to destroy it. In both cases, at the end of task the Formation is disbanded, and Mobiles return to normal work that is to explore the map.

A Maw can also start a War Task. As defined previously, War can occur when one of the fractions becomes stronger than all other fractions combined. When this is true, a single Maw can start communicating with other Maws using the *Ask For Alliance* message in order to create a force that could weaken the common Enemy Maw.

When a Maw is informed about an Enemy Formation, a Defend Task is initiated. This will cause all Mobiles to return home immediately to form a Formation of their own in order to destroy their (incoming) enemy.

V. Simulation

Now, let us describe the tested input values and the results of the preliminary simulations. The data set of our application was divided by harshness of the environment into two scenarios: *friendly* and *deadly*. Based on this division, two simulation sets were conducted, each returning different results. The comparison of said results leads to interesting observations regarding our main thesis.

In our opinion (hypothesis we started with, and tested in our simulations), cannibalism is a natural outcome of profit-driven logic, in which the only relevant factor in the decision making process is the amount of food that can be gathered as a profit, as it vastly increases the survival chances of an individual Maw.

This, however is not presented in the *Sandkings* story, where the survival of the race often had a higher priority than the survival of the individual. This leads to our hypothesis that with progressively decreasing friendliness of the environment, the aggression level towards Mobiles of other fractions, and other Maws, in particular, vastly increases.

In a friendly environment, Maws usually do not need to rely on Alliances, as they are strong enough, on their own, to cope with most threats in the terrarium. When the environment is much stronger than their respective power levels, cooperation – even if temporary – is the Maw's only chance of survival.

When the terrarium is filled with enough food and relatively weak Enemies, an endless simulation, i.e. prolonged lifespan of each race without clear victory of one of them – should be possible, as shifts in power balance between the fractions should be much smaller than within a deadly environment. This means that, most probably, with harsh configuration, a victorious Maw will quickly emerge.

A. Shared Data set

In the simulation we have used four Maws with their respective Mobiles, six different Food types, three Enemy types, and a simplified automatic spawning algorithm called the *Autogod*. The grid size was set to 50x50. This structure allowed us a simplified, yet visually engaging, representation of the environment described in the original novel.

Enemies' graphical size changed with time, accordingly to their health and attack level increased with time. The first was increased by the 4th square root of the tick count and the latter – by the 5th. There was 10% chance that an Enemy will be dropped on each God's step. In Tables I, II, III we present the input data sets that are constant in both Friendly (Section V-B) and Deadly (Section V-C) environments.

TABLE I
MAW PROPERTIES

Property	Value	Comments
strength	0	Strength at the beginning of the simulation.
food unit	0	How much food calories a Maw has at the start.
attack	0	Initial number. Changes depending on Maw's strength.
health	1000	Initial number. Changes depending on Maw's strength.
meat count	50	How much steak is dropped when it dies.

TABLE II
MOBILE PROPERTIES

Property	Value	Comments
steps per food	150	How much steps it can take until it starves.
stomach size	2	How much calorie at a time it can eat.
attack	5	How much damage it makes per one attack.
health	100	How much damage it can take before it dies.
meat count	1	How much meat is dropped when it dies.

TABLE III
FOOD PROPERTIES

Property	Weight	Calorie
Cabbage	1	10
Grape	3	30
Doughnut	5	50
Pizza	7	100
Meat	1	3
Stake	1	20

As discussed above, the Weight determines how many mobiles are needed to carry each food. Both properties – weight and calorie – are constant during the span of a simulation.

B. Friendly Environment Data Set

In the first simulation we have defined what we believed to be a friendly environment; one in which it is easy for the Mobiles to survive. Here, in the *Autogod* module, a 33% of chance is given that a food will be dropped on each of God's step. While the probability of spawning Enemies is equal in both environment types, in the friendly environment their strength and health is optimized so that the Mobiles should not have difficulty in eliminating them (see, Table IV).

TABLE IV
FRIENDLY DATA SET: ENEMY PROPERTIES

Enemy Type	Spider	Scorpion	Snake
Attack	10	20	40
Health	150	300	600
Meat value	1	4	20

C. Deadly Environment Data Set

This environment was set so that it would be hard for the Mobiles to survive. The purpose of this experiment was to observe how long they can live under harsh circumstances. In addition to this, the occurrence of wars and alliances in comparison to the Friendly Environment, was expected to change. The probability of Food being spawned by the Autogod has been dropped to 10%, which makes surviving much harder for the Sandkings. The Enemies' properties were also adjusted according to our vision of the harsh environment (see, Table V).

TABLE V
DEADLY DATA SET: ENEMY PROPERTIES

Enemy Type	Spider	Scorpion	Snake
Attack	15	30	60
Health	200	450	800
Meat number	1	4	20

VI. EXPERIMENTAL RESULTS

The following tables describe the results of 20 sample simulations (10 in friendly, 10 in harsh environment settings). Average values of each of the simulation types are presented. Note that we have not observed large variation in results. Henceforth, average values presented below are "representative" to both sets of experiments.

A. Simulation with the Friendly Data Set

As shown in Table VI, the application of the Friendly dataset (see Section V-B) resulted in simulations ending, on average, after 7971 ticks, spawning almost 40 Enemies and dropping 134.6 Food pieces on the map. The Maws have formed, on average, 3.2 alliances per simulation.

TABLE VI
FRIENDLY SIMULATION - GENERAL STATISTICS

GENERAL	Tick	Enemies spawned	Food dropped	Alliances
Average	7971.6	39.9	134.6	3.2
Per Tick	−−	0.005	0.017	0.00040

In addition to this, the statistics about the average, winner (last survivor) and losers are presented in Table VII. The columns describe the time when the death of one of the Fractions occurred, the number of lost Mobiles and the peak value of the Mobile count, and the amounts of Food consumed.

Finally, a third statistic concerning the number of Tasks (and by this, Formations) is shown in Table VIII.

TABLE VII
FRIENDLY SIMULATION - MAW STATISTICS

MAW	Death Tick	Lost	Max	Meat	Non-meat
Total AVG	2639.20	76.53	42.33	116.55	23.30
Per Tick		0.01	0.01	0.01	0.00
Winner AVG		89.13	51.25	136.88	30.63
Losers AVG	3650.56	73.52	38.11	103.11	19.59

TABLE VIII
FRIENDLY SIMULATION - TASK STATISTICS

TASK	Food	Creature	Defend	War
Total AVG	23.00	16.93	0.10	0.25
Per Tick	0.00	0.00	0.00	0.00
Winner AVG	41.75	27.88	0.13	0.50
Losers AVG	16.63	13.74	0.11	0.15

B. Simulations with the Deadly Data Set

In the Deadly Data Set – as shown in Table IX – the simulation ended, on average, after 3599 ticks, almost equally spawning 18 and 18.6 Enemies and Food pieces on the map. The Maws have formed 4.5 alliances per simulation.

This simulation run shorter than the friendly one, which means that the Mobiles faced a bigger difficulty level. The comparison of the alliance count also shows that, under these circumstances, the Maws are more likely to form an alliance to be able to combat a common enemy. Taking into account the difference between length of simulations and number of alliances made in Friendly and Deadly environments, in the first case alliances were made approximately every 2491 tick count and in the latter - every 800. This also means that when

facing harsh simulation, the Maws formed over 3 times more alliances.

TABLE IX
DEADLY SIMULATION - GENERAL STATISTICS

GENERAL	Tick	Enemies spawned	Food dropped	Alliances
Average	3599	18	18.6	4.5
Per Tick	−−	0.005	0.005	0.0013

The Maw statistics *(*see Table X) again show the time of death, lost and peak amount of Mobiles and the eating habits of the overall, winner *(*last standing) and losers. The lifespan per Fraction does not differ significantly from the Friendly Data Set, but the amounts of Mobiles and Food consumption are proportional to the harshness of the environment and resulting duration of simulation.

TABLE X
DEADLY SIMULATION - MAW STATISTICS

MAW	Death Tick	Lost	Max	Meat	Non-meat
Total AVG	2074.25	34.20	17.73	40.95	3.75
Per Tick		0.01	0.00	0.01	0.00
Winner AVG		26.50	26.00	53.50	5.20
Losers AVG	2765.67	36.77	14.97	36.77	3.27

In direct comparison, the Task statistics (as presented in Table XI) differ significantly. The Maws, when placed in a Friendly Environment, sent more of their offspring to fetch Food or to attack nearby Enemies, though this is most probably due to the scarcity of the Food and the strength of the Enemies.

TABLE XI
DEADLY SIMULATION - TASK STATISTICS

TASK	Food	Creature	Defend	War
Total AVG	3.10	5.13	0.13	0.25
Per Tick	0.00	0.00	0.00	0.00
Winner AVG	4.20	7.10	0.00	0.30
Losers AVG	2.73	4.47	0.17	0.23

The Deadly Data Set made the Maws focus their attacks *not* on the Creatures dropped into the Terrarium, but rather on the Mobiles of other Fractions as they were weaker and easier to eat. In addition, the winning Maw is more likely to have started a war – though this is only visible in the Friendly Simulation.

In the case of the Friendly Data Set, Maws died mainly due to starvation, where the winning Fraction would collect more than the other three. The Deadly Data Set showed some examples of Enemies *(*or, enemy Formations) attacking one of the Maws directly, leaving the defenders not enough time to prepare – leading to substantial losses or even death of the Sandkings tribes.

Most of the time, the first Maw, with the weakest start, died rather quickly in the simulation (as described in the book). The second Fraction to fall usually lasted until approximately 3/4 of the simulation, leaving the surviving two living mostly peacefully. In rare cases, observed in multiple simulations run during program development, all four died in a few hundred steps.

Alliances of three Maws against the fourth occurred very rarely, proportionally to the same strength growth for all Fractions. The Alliance system was most often used to attack Creatures in the Terrarium. A stronger Maw attacking one of the weaker ones, in order to gain Food, only happened rarely – when a Formation, which was sent out to attack a Creature, finished its task and started chasing another Fraction's Mobiles running into their Maw and killing her.

VII. CONCLUSIONS

Analysis of results from both Friendly and Deadly Data Sets led to surprising results. Our initial hypothesis was that, with the increased hostility of the environment, the Sandkings would develop cannibalistic behaviours and fight until eventually only one tribe is left.

Contrary to it, with harsher environment the Sandkings showed tendencies to cooperate in order to increase their own chances of survival, and refrained from attacking each other, as the sustained losses most probably would prove incomparable to the potential food gain. On the other hand in friendlier environment, where Sandkings amassed sizable amounts of food and grew in size and numbers, raiding other Maws proved profitable enough that acts of cannibalism have been observed.

Concluding, our hypothesis was faulty, as our Sandkings model proved that their aggressiveness increases not with harshness of environment they are located in, but with its friendliness. With the results presented, it is safe to state that the vision presented in work of George R. R. Martin was plausible, as Sandkings when facing extremely unfriendly conditions had to cooperate in order to survive.

ACKNOWLEDGEMENT

We would like to thank Marcin Paprzycki and Maria Ganzha for a fascinating introduction into this topic, wise guidance and constant kindness.

REFERENCES

[1] Martin, G. (1981). Sandkings. New York: Pocket Books.
[2] Collier, N. and North, M. (2015). Repast Java Getting Started. 1st ed. [ebook] Repast Developement Team. Available at: http://repast.sourceforge.net/docs/RepastJavaGettingStarted.pdf [Accessed 30.05.2015].
[3] Repast.sourceforge.net, (2015). Repast Suite. [online] Available at: http://repast.sourceforge.net/ [Accessed 30.05.2015].
[4] Railsback, Steven F., Steven L. Lytinen, and Stephen K. Jackson. 'Agent-Based Simulation Platforms: Review And Development Recommendations'.
[5] Crooks, Andrew. 'An Introduction To The Repast Software Recursive Porous Agent Simulation Toolkit'. 2015. Presentation.
[6] Schoonderwoerd, R., Holland, O., Bruten, J. and Rothkrantz, L. (1997). Ant-Based Load Balancing in Telecommunications Networks. Adaptive Behavior, 5(2), pp.169-207.
[7] Pratt, S., Sumpter, D., Mallon, E. and Franks, N. (2005). An agent-based model of collective nest choice by the ant Temnothorax albipennis. Animal Behaviour, 70(5), pp.1023-1036.
[8] Candelaria E. Sansores, Flavio Reyes, Hector F. Gómez, Juan Pavón, Luis E. CalderÃłn-Aguilera. BioMASS: a Biological Multi-Agent Simulation System. Proceedings of the Federated Conference on Computer Science and Information Systems pp. 675–682.