My life as a sim: evolving unique and engaging life stories using virtual worlds

Rubén H. García-Ortega[1], Pablo García-Sánchez[2], Antonio M. Mora and J.J. Merelo[2]

[1]Fundación I+D del Software Libre, Granada, Spain
[2]Dept. of Computer Architecture and Technology, University of Granada, Spain
raiben@gmail.com,pgarcia@atc.ugr.es,amorag@geneura.ugr.es,jmerelo@geneura.ugr.es

Abstract

Stories are not only painfully weaved by crafty writers in the solitude of their studios; they also have to be produced massively for non-player characters in the video game industry or tailored to particular tastes in personalized stories. However, the creation of fictional stories is a very complex task that usually implies a creative process where the author has to combine characters, conflicts and backstories to create an engaging narrative. This work describes a general methodology to generate cohesive and coherent backstories where desired archetypes (universally accepted literary symbols) can emerge in complex stochastic systems. This methodology supports the modeling and parametrization of the agents, the environment where they will live and the desired literary setting. The use of a Genetic Algorithm (GA) is proposed to establish the parameter configuration that will lead to backstories that best fit the setting. Information extracted from a simulation can then be used to create the literary work. To demonstrate the adequacy of the methodology, we perform an implementation using a specific multi-agent system and evaluate the results, testing with three different literary settings.

Introduction

In video games, Non-Player Characters (NPCs) are a type of characters that live in the game world to provide a more immersive experience and, in some cases, present a challenge to the human player. Modern Role Playing Games (RPGs), such as The Witcher™ or Skyrim™ include hundreds of NPCs. The effort to create a good interactive fiction script is directly proportional to the number of these characters. Thus, this kind of agents usually present limited behaviors, such as wandering in the villages, selling groceries or guarding the cities. Moreover, they usually offer scripted conversations (for example, to buy and sell objects to the player), or scripted behaviors, in which they interact with the player only if he/she conducts an specific action: for example, if the player steals something then a city guard would attack him. These characters are frequently programmed to interact with the human player, following a guided sequence of activities just with this purpose, so normally they do not interact among themselves. In a world with such a number of characters, their collective interactions could improve the gaming experience, leading to a richer and more immersive world. For example, hungry inhabitants could become thieves, guards could pursuit the thieves, villagers could fell in love with others, or different war alliances could emerge.

These facts have motivated us to create a first step to achieve this objective, developing a methodology to model the language, agents and literary setting to generate backstories. In this methodology, a set of probabilities and states are associated to agents' actions, and these probabilities are optimized by means of a Genetic Algorithm (Goldberg, 1989) to match with a specific literary archetype, defined by the fiction creator. The *archetypes* are behaviors and patterns universally accepted and present in the collective imaginary (Garry and El-Shamy, 2005). They allow to empathize with the characters and aid to immerse the spectator in the story (for example, the well-known *hero* archetype).

To validate our methodology we also present a multi-agent system called MADE (Massive Artificial Drama Engine) to model a self-organized virtual world where every element influences each other, following cause-effect behaviors in a coherent manner. This system must be a suitable environment for the plot of a specific literary work, also being interesting for the player/spectator.

In this work we investigate whether it is possible to model a virtual environment inhabited by hundred of characters with interesting auto-generated behaviors based on literary archetypes. Also, we test if the personality of the agents can be parametrized to obtain these different emergent behaviors. A GA will be used to find these adequate parameter values that allow the creation of quality sub-plots.

The rest of the work is structured as follows: after the state of the art, the proposed methodology is presented. Then, a complete example of application of the methodology is explained, including the experimental results for three different literary settings. Finally, conclusions and future work are discussed.

State of the art

Auto-generated interactive fiction research is mainly focused in methods to create the process of a story generation

(Nairat et al., 2011). Story generation, or storytelling, can be divided in two areas: interactive and non-interactive. In the first one, and according to (Arinbjarnar et al., 2009), an *interactive drama* is defined in a virtual world where the user is free to interact with the NPCs and objects in an interesting experience (from the dramatic point of view), which should be different in each execution, and adapted to the user's interaction. The generation of interactive dramas can be based on a script structure (Young et al., 2004) and dramatic structures, where an inciting accident provides the motive of the drama, is followed by an increase of complications to a climax point and, finally, descending to a closure. Two examples of this approach are The Oz Project (Sloane, 2000), that requires generated narratives to follow a dramatic arc, and Façade (Mateas and Stern, 2003), that uses a structure which they call neo-Aristotelian, an adaptation of the Aristotelian structure to interactivity.

On the other hand, in *non-interactive plot generation systems* the user does not take control as the protagonist but can participate in the final result. For example, in the system presented by Pizzi et al. (2007) the user can change the emotions of the other characters but as an spectator not as an actor.

Those techniques and systems presented at the moment are focused in generating stories (interactive and non-interactive), but the present methodology, as opposed to this concept, is focused in *generating a setting* because its aim is the massive background generation for secondary characters, in order to provide a context for the writer and the player to perceive a virtual world as coherent, detailed and enriched.

The narrative is addressed by our methodology as the final step, giving freedom to creators. This issue has been studied in the systems presented in the survey by Arinbjarnar et al. (2009). Works in their survey define the plot as as something that emerges from the behavior of the agents that follow a set of rules. In the proposed methodology, the agents' behavior is produced by their personality and the environment. That is, the agents do not follow a plot, but they generate the plot itself. Furthermore, the works of the survey generate plots in worlds with a limited number of characters. This restriction does not exist in our methodology, where the number of characters to create is unlimited: they are created massively and their goal is to relate in a complex system and generate backgrounds that fit the setting of a plot, not the plot itself.

The present methodology follows the ideas of the work by Epstein and Axtell (1996), the first widely known multi-agent generative social model. As a step of the methodology, a self-organizing system if defined here following the methodology introduced by Gershenson (2005): a virtual world, agents who are born, grow, interact, reproduce and dead; resources (food), mediators, and relations of rivalry (friction) and cooperation (synergy). In other step of our methodology, the actions of the agents are parametrized according the work of Nairat et al. (2011), based in the use of GAs in order to obtain a plot (solution) where two characters interact in a creative way.

The proposed methodology is innovative since the goal has not been addressed before. Previous researches are focused on the plot, the interaction, and the narrative, but our proposal is focused in backgrounds (not in plot), where different archetypes emerge involving a starting point for future plots and subplots.

According to the taxonomy described by Togelius et al. (2011), the present methodology can be classified as a *procedural content generator* (*PGC*) mainly related to *optional content*, with *stochastic generation* and modeled as a *generate-and-test* algorithm (search based), that performs the optimizations of the process during the game development (*offline*).

Methodology

The mood, or the atmosphere, of the literary setting is one element in the narrative structure of a piece of literature. It is established in order to affect the reader emotionally and psychologically and provide a feeling for the narrative.

Our methodology defines different steps that will lead the user from the initial question "*How could I obtain secondary characters for my setting, with coherent backstories consistent with its mood?*" to a system that can automatically generate a setting massively populated with characters and backstories, where different desired behaviors or archetypes emerge from their interactions.

In our approach, the mood of the setting will be modeled as a group of abstract archetypes and different conditions over them. These archetypes, conceptually modeled, would be designed and instantiated as regular expressions over a language used to describe the backstories along with the social relationships between them. Those instantiations will be used by the GA to obtain the fitness of each solution. The process is iterative and presented as a waterfall model, where each step influences the following one.

The methodology includes the following steps, as shown in Figure 1: Modeling (the language, the literary setting, the agent), Definition of the GA characteristics, Instantiation, Execution and Interpretation, which will be exposed in the following subsections.

Modeling

The Modeling step affects different elements that make up a system where different agents generate words, lately interpreted as backstories, whose symbols belong to a language that also defines the desired archetypes.

Modeling of the language Each agent has to be modeled as a Finite State Machine (FSM) (Booth, 1967) whose transitions generate symbols in a language based on the following one, expressed in Backus-Naur Formalism (BNF):

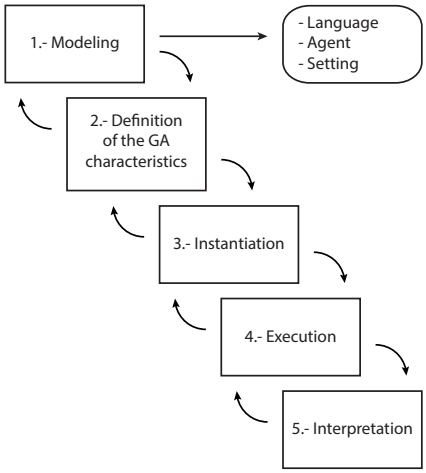


Figure 1: The proposed methodology for designing literary backstories

```
<backstory> ::= <action_line> [<backstory>]
<action_line> := <date> <action>;
<action> = <action_id> [<direct_object>]
           [to <indirect_object>]
```

Starting from this partial grammar, the user has to define the expressions:

- <**date**>: Should be expressed in a way that is meaningful to the type of background that we are looking for.For example, if we are talking about persons, it could be the age of the agent expressed in days.
- <**action_id**>: Different actions can be taken into account, and will depend on the kind of archetypes we will look for. For example, if we are trying to find archetypes about love like the classical Romeo and Juliet one, where two characters are in love but belongs to families in war and finally ends up with their death, we could include actions and relations like “loves”, “hates”, “is son of”, “is daughter of”, “is parent of”, “die”, etc.
- <**direct_object**>: The possible values of the direct objects depend on the nature of the actions.
- <**indirect_object**>: Should define the unique id of an agent or a group of agents.

The backstory generated by each agent using the language modeled this way will be automatic and human readable. Eventually, the backstories will evoke archetypes and behaviors. As we will see in the next sections, we will try to promote the emergence of specific archetypes in order to retrieve the best backstories for our setting.

Agent modeling In a top-down approach (the opposite to the one proposed in the current methodology), if the creative needs a second character or an extra for a story, he or she creates it out of the blue, with the background that better fits with his/her needs. This approach, valid for a small amount of characters and poor requisites of coherence between them, becomes more and more complex when we add more characters to the environment, because each addition implies new relations in their social network.

Otherwise, the bottom-up approach promotes the agent as one of the pillars of the system because it offers the coherence of the backstory regarding to the timeline (for example, an agent cannot die before he or she is born). Some actions need to be sequential and to the relations (many actions involve two different agents). An agent is just a simple tool to generate a background, a character. In an environment where many characters can “live”, make decisions taking into account the other characters and affecting them, the generated background becomes complex, many links have been created and no creative process has taken part on it (the creative process is present in the definition and modeling of the agent itself).

Given this premise, a multi-agent system seems promising for generating this kind of complex relations and backstories.

An agent can be seen as a Finite State Machine (FSM), so the multi-agent system relies in the sequential executions of time portions. Every execution implies a review of the current state and (maybe) one or more actions done, depending on local and external properties.

The agent should contemplate different states depending on the type of information we are interested in (“alive”, “dead”, “pregnant”, etc) and should also have local properties that define the possibility to make certain decisions.

Agents live in an *environment*, understood as the virtual spatio-temporal frame where the agents play their lives. This environment is also responsible for executing the main process of each agent (corresponding to a portion of time). Each iteration the agents should be chosen randomly or based in classical role-playing games features like “initiative”. Also, the environment provides a set of functions to interact with other characters and the environment itself.

The key in the use of agents to create background for characters relies in the following fact: Some actions are performed statically depending on the inputs and the current state but other (the majority) also depends on agent’s local features and probabilities. Even if all the agents share these values, it is difficult to predict the result when the system is so complex. A minimal change in one probability to make one decision can imply completely different scenes. For this reason, some actions should be statically defined and others have to depend on initialization properties. We define two kind of properties:

- **Base properties**: Those that are intrinsically defined by the nature of the conceptual agent. For example, if we

are modeling a simplification of a person, the average life expectancy could take values from 70 to 85 years. An agent that lives 100 years would be unusual, and an agent that lives 200 years should not be possible.

- **Searchable properties**: Those used to increment or decrement the base values in the established limits or those that are used directly as probabilities to make decisions.

Due to the complexity of the system and the introduction of probabilities, even if all the agents have the same features and probabilities, the sequence of the actions for every agent becomes unpredictable. For example, if we have our agent modeled and we choose random values for the configuration, roughly, the characters' backstories will be "normal", showing more or less the stereotyped behaviors (that make the character a group representative rather than an individual), but maybe, some of them show higher level non-modeled behaviors. In the next subsection, we will provide a way to model those high-level behavior in the way that, given a executed environment, a computer is able to find them and, in the Execution step, obtain a configuration that promote the apparition of these archetypes in the execution.

Literary setting modeling The setting of a story is defined as the historical moment in time and geographic location in which it takes place, and helps initiate the main backdrop and mood for a story. Conceptually, the setting of a story is, in this methodology, independent from the agent design, but has to agree with it in the selected language: The agent generates actions in a previously defined language with a clear semantic interpretation. On the other hand, the setting is composed by criteria over patterns for this language and the social networks derived from its usage. In other words, the agents create backstories and the settings matches specifics patterns inside these backstories.

For example, if we have a medieval storytelling setting and we have defined a language where the actions "love", "hate", "attack", "defend", "win" y "lose" are used, we could try to find classic archetypes like the "villain" and the "hero". A villain could be a character that hate many others, attacks them and win. A hero could be a character who is loved by many people, that eventually defends them and then attacks to the villain. Moreover, if the mood of the story is sentimental, we could be interested in a "heartbreak and meet again" archetype, where two characters are in love, after that they hate each other and finally fall in love again.

In our methodology, an archetype is designed as a function that receives the backstory of an agent, processes it, tries to find patterns and interpret the social network related to the agent, and returns a *true* value if the agent matches the behavior. The complexity of the agent is not evaluated, just the result of its execution in this environment. It is important to remark that the agent has to be modeled to play a normal life (stereotype), and that the archetypes work as high level behaviors not directly implemented.

A setting can be seen as a function over the agents that matches an archetype. If the archetypes emerge in the desired way, the setting function would return high values.

In the next section, we will explain the mechanism that will optimize the agent's searchable features to obtain backstories coherent with the mood of the setting.

Features of the Genetic Algorithm

The codification of the chromosome of the individuals of the GA consists in mapping the searchable properties to an array of values.

The fitness function of a chromosome is the result of calculating the average of the application of the setting function over n executions of the environment with defined base properties (where n is the minimum number that reduces the error and has to be calculated empirically).

It is important to clarify that the term "individual" in this work refers to a solution in the GA that models a whole environment, where a number of different "agents" are living. So, an "individual" in the GA is not equal to an "agent".

At this point, we remark the possibility of using different number of *profiles*. In this context, a profile is a set of properties assigned to an agent. If two agents have different profiles, their behaviour in the face of the same inputs can be different. In the chromosome, the use of n profiles means a chromosome size equal to the number of searchable properties multiplied by n.

Intuitively, increasing the number of profiles will lead to richer backgrounds, but *a priori*, since the system is complex, it is very difficult to establish which number of profiles will lead to the fittest solution without empirical tests. In some cases, a small number of profiles can generate many different archetypes and the other way around. Moreover, it is important to remark that the bigger the chromosome is, the slowly the solution converge.

Instantiation and execution

After the Genetic Algorithm is configured, some properties need to be established in order to fit to the desired setting:

- Base properties: Including the threshold for the agent's features, and environment parameters (i.e. size of the world, resources, etc).
- Genetic Algorithm parameters: selection, genetics operators (crossover and mutation and termination condition)
- Number of profiles

Once the parameters for the GA, the environment, the agent and the number of profiles are set, the GA can be executed. In some cases, the fitness can be improved by modifying base parameters, the evaluation of the archetypes or the logic of the agent. The process is iterative and a fine

tuning is essential to obtain the best fitness. Once the best solution has been found, the environment can be executed and the backstories generated can be used.

Interpretation

Due to the nature of the grammar proposed for the actions, they can be directly converted to natural language. Applying the setting function (used as fitness by the GA) to the executed environment could be useful to tag the backstories with the different archetypes found and their explanations. As an example, if our setting is the "far west" and we need non-player characters to populate a saloon, we may want to find a "Billy the kid" archetype and three "player" archetypes. If they are met by the main character he/she may discover their backstories, that would be coherent with the world they live in and the mood of the story.

Applying the methodology

To validate our methodology we apply its steps using a specific scenario and environment. In this scenario we want to model the next story: - "A number of rats live, eat, reproduce, compete for food and death within the walls of the Invisible University of Ankh-Morpork[1]. As the University professors, these rats are very vindictive and territorial."

Agents in the MADE environment

The first step is to model the agents and their environment. In this work we propose the MADE (Massive Artificial Drama Engine for non-player characters) environment, a virtual place where different agents play their artificial lives. Its functions are to initialize agents in the map, control the time, execute the agents during a time unit (for example, a day in the story) and update the map.

The MADE environment can be configured by using the following parameters (with the values that will be used in the experimental section in brackets): Number of agents initially placed (15), map square grid dimension (10), number of rations randomly placed in the grid each day (10) and duration in virtual days of the execution of the environment (1000). These parameters can affect directly to the behavior of the agents.

A MADE Agent lives in a MADE Environment, occupies a cell in the grid, moves around looking for food or mates and interacts with other agents.

The design of a MADE Agent has some restrictions:

- An array of parameters (probabilities represented as real numbers between 0 and 1) should be provided in its initialization. These parameters should be used by the agent in its day-by-day decisions.

[1]This is our tribute to writer Terry Pratchett, whose books have inspired us to create these agents.

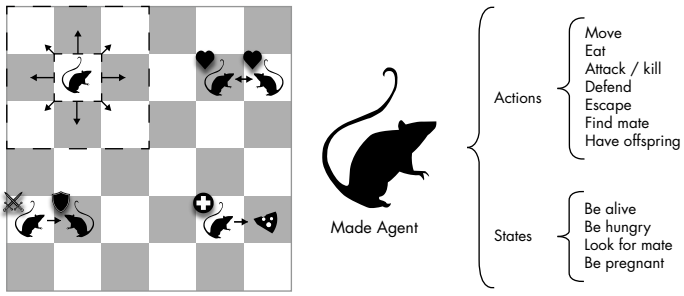


Figure 2: Actions and states modeled in the MADE Agent.

- Every action is subject to be logged for later analysis.

- As a result of some iterations of the agent's day-based life-cycle, the agent should present the behavior of a *living thing*, that is born, eats, grows, reacts, reproduces and dies.

A very simple agent has been designed for this study: a virtual rat, that models:

- Four states (be alive, be hungry, look for mate and be pregnant) that represent internal situations that will lead the agent to perform the actions described in the item bellow.

- Seven actions (move, eat, attack, defend, escape, find mate and have offspring) that lead to a basic instinctive animal behavior, very useful for this work since it can be the canvas of complex *humanized* behavior patterns.

- Parameters that define the characteristics and probabilities to perform actions depending on the state.

It is important to remark that no "feelings" and no "memory" have been modeled in the MADE agent for this study. This is illustrated in Figure 2.

Every decision made by the agent is based on its state and its characteristics (probabilities to perform different actions).

The MADE Agent is created using twelve parameters, that define its base features and probabilities to make the decisions presented in the state diagram in Figure 3. The execution of an agent is dynamic, and depends on the internal probabilities and states but also on the neighborhood, and the map configuration. Even so, we can say that these initial parameters define in some way the possible situations where the agent could be involved.

The source code of the MADE environment and the algorithms used in this work are publicly available in `https://github.com/raiben/made` under a LGPL license.

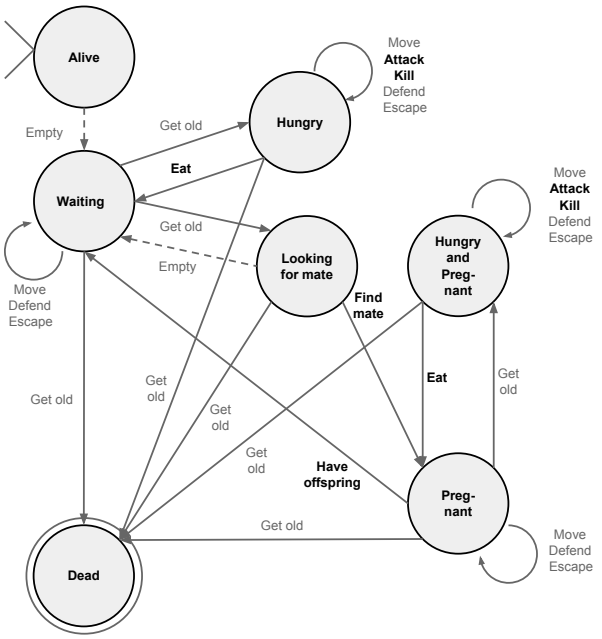


Figure 3: MADE Agent's finite state machine

Definition of the literary setting

As previously said, agents are independent of the desired literary settings. To validate our approach three different literary settings are going to be tested.

The first (and simpler) literary scene is called "revenge" and its goal is to model individuals with complex memory-based behaviour between two characters (although the agents in our environment have not memory). It tries to find the number of profiles and values which are optimal to make *revenge* archetype emerge in as many agents as possible after 1000 days. An agent (a) will be considered as an *avenger* if it has been attacked by other agent (b) and after that, in a moment in its life, it has satisfactory attacked the agent b, in *revenge*. The value of the days is set to 1000 because is a duration long enough to make the archetype emerge.

The secondary literary scene, "territorial war", aggregates different sample archetypes where many factors must be taken into account. It tries to find the number of profiles and values that generate at the end of the run an equal distribution of the archetypes *downtrodden* (an agent that has been attacked at least two times and has defended the position), *warrior* (if it has satisfactory attacked at least five times), *helpless* (if it has been attacked at least ten times and has not defended the position) and *bad warrior* (if it has been defeated at least ten times).

Also, after 1000 virtual days, the alive population will be the 60% of the total population (archetype *survival population*). We have used the presented values to define this scene because, in our opinion, they model an interesting literary setting.

Finally, the last literary setting, "Shakespearean", models different archetypes present in the works of Shakespeare to create a richer and complex scene. In this scene, there exist several well-known archetypes, proposed in the work of Vogler (1998). The first one is *star crossed lovers* represents the Romeo And Juliet: rats who have offspring and whose parents have participated in a fight. Also, other archetypes must arise in this setting: the *shadow* (an agent that kills another), the *hero* (an agent that defeats a *shadow*), the *mentor* (an agent that gives food to the *hero*) and the *threshold guardian* (an agent that attacks the *hero* at least two times). The last two archetypes can not be *hero* or *shadow*, adding complexity to the environment.

Definition of the genetic algorithm

As previously said, the parameters used to define the agents are mapped into a chromosome, and a Genetic Algorithm is used to evolve the solution. The fitness function (or setting function) is expressed in terms of:

- **Regular expressions applied to the log of each agent in the environment:** An agent is tagged when a regular expression matches its log.
- **A numeric function over the number of tagged agents for each archetype:** the fitness of the solution is incremented with the returning value.

Different fitness functions have been used. In the first literary setting ("revenge"), every agent whose log matches the archetype adds one point to the fitness. Therefore, the goal is to recreate an environment where several "avenger" agents exist.

To model the "territorial war" setting we have defined the fitness function as follows: if the exact percentage of agents are tagged with one archetype defined, 1 point is added to the fitness. The maximum is therefore, 5 points. However, all the fitness values use a normal distribution over the percentage of appearance. For example, with the archetype "survival population", the maximum value (1) is obtained when the 60% percent of the population is alive, and the normal distribution begins in the 30% and ends in the 90%. For the rest of the archetypes, 1 point is added if the 22'5% of the population is tagged with each one of the archetypes, and each normal distribution begins in the 8% and ends in the 30%.

Finally, the function for the "Shakespearean" scene is the sum of the proportions of each archetype with respect to all the population. Therefore the maximum would be 5 (if the archetypes were not exclusive).

Thanks to the agents' logs, we can know every event (internal and external) of their lives, and evaluate their interest or adequacy to a specific literary setting.

Parameter	Value
Codification	12 alleles per profile
Fitness function	Average of 10 executions.
Natural selector	Original Rate: 0.9
Crossover operator	Rate: 35%
Mutation operator	Desired Rate: 12
Stop condition	100 generations
Population size	30

Table 1: Parametrization of the Genetic Algorithm

In this work, we have implemented a method based in regular expressions with backreferences. The proposed technique puts annotations in every agent whose log matches a complex regular expression able to find emerging high level behaviors, not implemented in the life-cycle. In this case, we do not exactly know how many roles are necessary to create a world of "warrior", "vindictive" or "shadow" rats, so different number of profiles (from one to five) will be considered.

If only one profile is used in a run, all the agents are created with the same parameters, evolved by the Genetic Algorithm. If more profiles are used, they are assigned to the agents in order of appearance in a loop. Our assumption is that some archetypes could emerge using one profile and other will need more (those that require two clearly differentiated roles). It is important to remark that the number of alleles of the chromosome are multiplied by the number of profiles, so the convergence of the solution will be be affected by the number of profiles used.

Execution and results

For the experiments performed in this work, we have used the parameters shown in Table 1. These values have been chosen empirically after several test runs.

The results of the experiments are shown in Table 2. It shows the average of the best fitness and the average population fitness at the end of 30 executions for each configurations. Boxplots of the best fitness obtained are shown in Figure 4.

With respect to the first literary scene, the Kruskal-Wallis and Wilcoxon pairwise comparison shows significant differences among all configurations (p-value $<<$ 0.05) except between P2 and P3 (p-value=0.3). Therefore, we can conclude that in this kind of global archetype only a profile must be used for obtaining the best results. This makes sense, because we are looking for one type of local archetypes (*avenger*), so adding extra profiles leads to different behaviors of the agents.

The results of the second literary setting ("territorial war") shows differences among all the number of profiles (p-value $<<$0.05). As we suspected, it is clear that using one profile is not enough to emerge the desired archetype. However, the pairwise comparison using Wilcoxon does not find significant differences using more than 2 profiles. This can be explained because an agent could share more than one archetype at the same time. A promising number of profiles could be 4, because their only lower outlier is not as distributed as the others.

Profiles	Setting 1	Setting 2	Setting 3
1	495.513 ± 20.091	0.765 ± 0.037	2.584 ± 0.044
2	471.206 ± 24.550	1.063 ± 0.115	2.433 ± 0.145
3	455.42 ± 28.240	1.093 ± 0.063	2.336 ± 0.141
4	431.926 ± 31.682	1.084 ± 0.048	2.281 ± 0.052
5	411.24 ± 25.023	1.045 ± 0.110	2.266 ± 0.068

Table 2: Results for 30 executions of each configuration using 1 to 5 profiles (average best fitness $\pm$ std. dev).

Finally, the results of "Shakespearean" scene surprisingly show that using only one profile is the best choice to generate a complex world as the one desired, decreasing with the number of profiles (except for 4 and 5, where no significant difference exist, with p-value$>$0.05). This can be explained because, to model all the archetypes, all actions available in the environment should appear. Therefore, a profile optimized to trigger all available actions would generate more archetypes. Also, and as previously said, using more profiles requires more time to converge, as we are using larger chromosomes. Furthermore, the actions of our environment could be quite simple to model complex stories with different personalities.

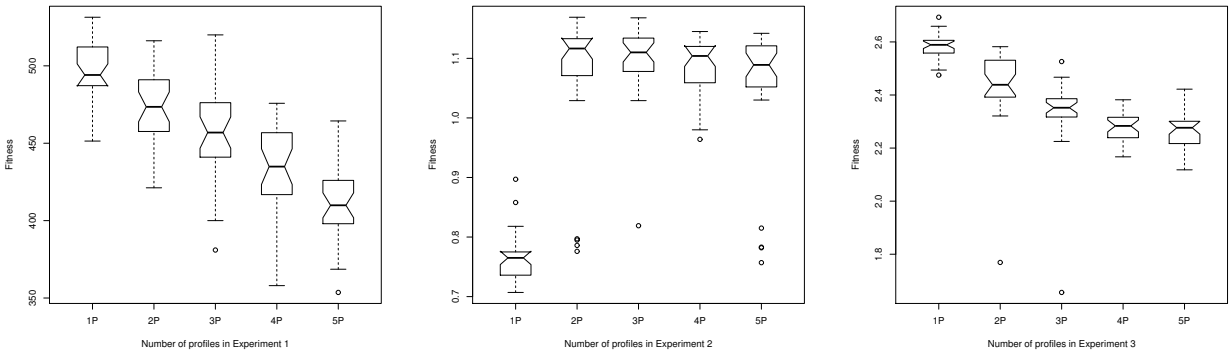
Convert results to literary language

Every agent has a log that stores all the relevant events in its life in a simple format. Each line of the log indicates the day, the event in a short readable format and some extra information. Every agent's log in the MADE Environment is coherent with all others' logs. Many plots are being created, with a structured format that can be read by game engines or natural language processors. This log can be used for evaluation, for example, obtaining feedback from human readers.

Conclusions

This work presents a general methodology to design emergent literary stories in massive virtual worlds. The described steps include the modeling of the agents and literary setting. Then a Genetic Algorithm is used to optimize the parameters of the agent's profiles (behaviors) using as fitness a function that models the literary setting.

Given a literary setting, an author of a story or a video game could define different rates of archetypes or behavior patterns, and use the techniques described in the present work to obtain the optimal profiles. In this work, several profiles have been generated in the MADE (Massive Drama



(a) Setting 1 (b) Setting 2 (c) Setting 3

Figure 4: Average fitness of the 30 best individuals of the GA for each setting.

Engine for non-player characters) Environment to produce a background (or set of characters' lives) where the archetypes have emerged and have automatically created massive backstories coherent with the settings of the artwork.

In future works, more complex agents will be used, with different rules to be modeled. For example, we plan to model more human behaviors such as love or envy, to generate interesting plots such as wars, weddings, or family crimes. More different fitness functions will be tested, for example, taking into account human opinions to establish the interestingness of a generated plot. Also, this system will be tested into an existent and well-known game, such as Skyrim™, whose AI engine is publicly available for players and researchers.

Acknowledgments

This work has been supported in part by FPU research grant AP2009-2942 and projects SIPESCA (under Programa Operativo FEDER de Andaluca 2007-2013), and TIN2011-28627-C04-02.

References

Arinbjarnar, M., Barber, H., and Kudenko, D. (2009). A critical review of interactive drama systems. In *AISB 2009 Symposium. AI & Games, Edinburgh*. Citeseer.

Booth, T. L. (1967). *Sequential Machines and Automata Theory*. John Wiley and Sons, Inc., New York, 1st edition.

Epstein, J. M. and Axtell, R. L. (1996). *Growing Artificial Societies: Social Science from the Bottom Up (Complex Adaptive Systems)*. The MIT Press.

Garry, J. and El-Shamy, H. M. (2005). *Archetypes and Motifs in Folklore and Literature: A Handbook*. M.E. Sharpe.

Gershenson, C. (2005). A general methodology for designing self-organizing systems. *arXiv preprint nlin/0505009*.

Goldberg, D. E. (1989). *Genetic Algorithms in search, optimization and machine learning*. Addison Wesley.

Mateas, M. and Stern, A. (2003). Façade: An experiment in building a fully-realized interactive drama. In *Game Developers Conference, Game Design track*, volume 2, page 82.

Nairat, M., Dahlstedt, P., and Nordahl, M. G. (2011). Character evolution approach to generative storytelling. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 1258–1263. IEEE.

Pizzi, D., Charles, F., Lugrin, J.-L., and Cavazza, M. (2007). Interactive storytelling with literary feelings. In *Affective Computing and Intelligent Interaction*, pages 630–641. Springer.

Sloane, S. (2000). *Digital fictions: Storytelling in a material world*. Greenwood Publishing Group.

Togelius, J., Yannakakis, G. N., Stanley, K. O., and Browne, C. (2011). Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186.

Vogler, C. (1998). *The writer's journey*. Wiese.

Young, R. M., Riedl, M. O., Branly, M., Jhala, A., Martin, R., and Saretto, C. (2004). An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development*, 1(1):51–70.