Planning with Dynamic Noise for Emergent Storytelling

David B. Carvalho
Erick B. Passos
Esteban G. Clua
Computer Science Institute
UFF - Federal Fluminense University

Abstract

Since the inclusion of stories in games, developers have been working on making them respond in accord to the player's options. However, make dynamical stories is not a trivial task. Applied to the context of games, it is necessary to control the characters and the environment without binding them to a single development and at the same time assuring interesting ends. This paper presents a work in the field of emergent stories with the objective of create more alternative developments and make the characters act more naturally by inserting and controlling errors, here called noise, in their beliefs of the state of the world. Here we make the characters loose or change some information and take wrong decisions. This work utilized one act of the Little Red Riding Hood story in a turn-based fashion as test case, and obtained interesting results in the story generation step.

Keywords:: Noise, Storytelling, Planning, Story Generation

Author's Contact:

{dbatista,epassos,esteban}@ic.uff.br

1 Introduction

Stories, commonly modeled as a finite state machine paradigm in games, are beginning to be worked as a more participating and dynamic component by reacting and adapting to the interventions of the player. However, these interventions not necessarily lead one story to a desired end. The answers to how much one player should interfere with the events of a story and how much the story should respond to the player's actions divided the study of storytelling in two approaches. The plot-based approach [Pozzer 2005] works with pre-built plots that guide the events into predetermined situations, giving fewer possibilities of interventions but ensuring a full developed story. The character-based approach [Cavazza et al. 2002] is modeled in order to give more freedom to the characters, enabling a story to emerge from their actions. This may lead to stories that respond better to the player's actions, but with no guarantees that they will reach a predefined conclusion.

The characters' freedom is related to the possibility to choose what actions to take towards their own particular goals. Some approaches [Charles and Cavazza 2004; Porteous et al. 2010] use these two key features (actions and goals) to make the same story take different courses. The characters may lead one story to more than one end if they can perform different sets of actions in a given situation. The handling of goals may also influence one story by giving a chance to one character to reach his own set of goals in any order. In any case, the generation process will be limited to the combinations of all actions and/or all goals. Also, for one character to act more naturally, it is necessary to reduce his options to the ones coherent with his behavior. As more options are interesting to enable a storytelling system react properly to the player, we believe that more techniques are needed to give alternatives to the generation process and make the characters act realistically.

The simulation and treatment of errors is a common study in the field of Artificial Intelligence [Klein and Dellarocas 1999; Broverman and Croft 1987], because the use of agents in real situations require them to consider fails in their sensors and actions. The story generation process could also benefit from the use of errors by inserting them into the characters' knowledge and leading to wrong actions on purpose. The occurrence of unequal errors in different executions of the same story can make one character choose different actions. This increases the number of development possibilities. Some stories even utilize this resort to lead their characters to create a problem and then make the rest of the events a series of attempts to solve it.

This work proposes the modeling and insertion of noises in the knowledge base of the characters, with some control over what changes are done. Work with noises draws at least one problem: how to control them without compromising the knowledge necessary to make the story happen. The solution of this problem in a general way is not in the scope of this work, but we describe how it was treated in our test and how it can be done in similar cases.

This paper is organized as follows: section 2 relates this research to previous ones in the field; Section 3 explains the architecture of the system created for this work; Section 4 details the noise insertion technique dividing it in two main approaches; Section 5 describes one test case scenario and its results; and finally in section 6 we conclude this paper with an analysis of the outcome of the generated stories and propose some future work.

2 Related Work

Planning techniques have been developed and used together with several approaches in the context of storytelling [Cavazza et al. 2002; Kruizinga 2007; da Silva 2007]. The use of each technique is directly related to the specific requirements of the system, to the level of user interaction and control over development possibilities. For this work, we choose a GraphPlan algorithm for the planning step (details about the Graph planning can be found in [Russell and Norvig 2010]), but it would be interesting to test the use of noise with different techniques and see what kind of variations they would generate with this feature.

Cavazza [2002] describes a system which utilizes characters whose actions are planned through their own HTN networks. Each network disassembles the characters' goals in a set of smaller ones to which are given different ways to achieve. In [da Silva 2007], the author also worked with HTN networks, but in a plot-based approach and inserting non-deterministic effects in the characters' actions. In this approach, even different effects must lead to predetermined ends, so the number of alternatives is given by the number of acceptable stories these consequences may generate. Mateas and Stern [2005] developed FAÇADE, a game that uses a hybrid of the character and plot approaches. The characters are controlled by a drama manager that selects pre-built situations, called beats, to make them act in accord of their personalities and goals. The generation step creates stories that are tied to the combination of these beats. Our work tries to generate different developments for stories not by giving options of actions or alternative events for the same situations, but by the management of errors in the characters knowledge, which can be considered a supplementary feature for those works.

Even with the generation of alternatives events as the main objective of the application of noise, this technique also lead the characters to act more naturally, because it simulates knowledge errors, that may be considered a common problem with characters. In this field there are several works such as [Damiano and Lombardo 2009; Su et al. 2007] that simulates the emotions and values to guide the characters' actions, and [Bosse et al. 2007] that works with the use of knowledge of one character about the others to, for example, make him influence them to reach his own goal. We consider these approaches useful to make stories more credible and consequently more immersive.

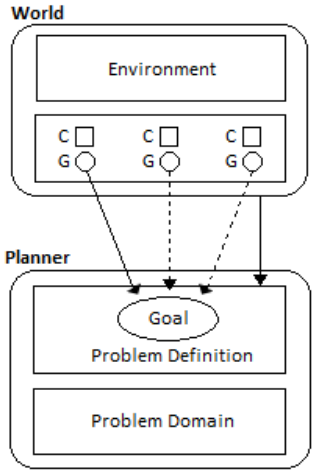

Figure 1: *Representation of the system architecture. It works with a planner and a world state composed of an environment, a set of characters and their goals.*

3 System Architecture

The proposed system works with a set of characters, an environment and a planning algorithm. The characters are described with a collection of attributes that compose their state and the environment with a set of locations and paths between them. These descriptions together form the world state. Each character, to guide his actions also has his particular goal. The planning algorithm receives as input the world state and the goal of one character at a time, generating a plan for every character in a turn-based fashion. Every character in his own turn executes the first action of his plan, then the system follows with the next character. Figure 1 shows this architecture.

Every character has its own unique goal which may interfere with the others', like kill another character or may only concern his own state like obtain a determined item. As the characters are not able to assume new goals, the conflicting ones are the criterion for the end of the system execution. If there are no conflicting goals, the system may cease its execution when they are all resolved.

To support independent characters that follow their own goals based in what they know about the environment, the planning algorithm is used to generate a plan for each character at a time, in a turn based fashion, by taking his goal as the state to be reached. The plan generated in every turn only represents one solution for the current state of the world. Taking into account that every character, in its turn, make one action towards the goal, the world state changes in every turn, so the characters must replan their actions every time they make one move. The system proceeds following the described cycle: the first character plans its actions and executes the first one returned in the plan, the second one makes its own plan already considering the changes done by the first, and so on until all the characters make one action. Code 1 shows the turn-based fashion.

```
Input:  A set of characters and an environment.
Output: A story composed of a set of actions.

planner.characters = characters;
planner.map = map;
end = false;
while (!end)
  for each char in characters
    planner.character = char;
    plan = planner.makePlan();
    action = first action of char in plan;
    updateWorldState(action);
    story.add(action);
    if (reached end condition of story)
      end = true;
      break;
return story;
```

Code 1: *Turn-based Story Generation*

The goals used by the planning algorithm, given the world state and the possibilities of state changes of the environment, must be reachable or at least it must be possible for the algorithm to consider a goal as unreachable, otherwise the system will execute indefinitely. The only case in which it is acceptable for a goal to become unreachable is when its conditions are made intangibles by the fulfilling of a conflicting goal, which will lead to the end of execution. One goal may not be made impossible by the use of noises.

The use of a turn-based planning has the disadvantage of not being able to execute more than one action in parallel. However, the characters do not need to consider changes in the ambient while they are acting in one cycle.

The system uses PDDL (Planning Domain Definition Language) version 3.0 together with Saigol's [2007] implementation of the GraphPlan algorithm.

4 Noise

The character-based approach works with the development of events without a pre-built plot, based only in the description of the world and of the characters. This description must be detailed enough to enable one character to make decisions, express his personality and show possibilities of story variations. The problem with this approach is that increasing the complexity of the environment and the amount of actions per characters not necessarily generates all the interesting types of situations in which a character may be and it not necessarily will make the same character, in two different executions, take different decisions at the same point of story, because of the determinism intrinsic to a formal agent description.

Another way to insert new possibilities for the development of a character-based story is to make the characters receive the information from the environment with some noise. The noises may be an interesting font of changes because they can make a character take wrong decisions and follow them until realizing that the consequence of his actions are not coherent with his goals, allowing the occurrence of unexpected events that may accumulate and lead to interesting ends. Even this approach alone may not be enough to generate considerable changes in stories, but combined with other factors (such as the usage of different actions), can increase the number of different developments.

In our system, the noise is inserted in the characters perceptions of the world and can make some information disappear (omission noise) or change its content (divergence noise).

4.1 Omission Noise

The omission noise is characterized by the removal of some of the information that was received by a character. It may be removed from the perceptions of an agent at the moment it uses its sensors, or from the current knowledge-base. In this system case, the omission noise is used in the environment description to remove some information from the original graph. The planning algorithm receives a character and the environment, in each turn, modified with the removal of some information. This way a character is forced to replan its actions because of the movements done by the others and because information used to make the current plan may be missing.

Considering, as example, a world composed of interconnected rooms that may or may not have some items, the omission noise algorithm could make some items or edges disappear from the problem description. A character that has to collect some of these items may become obligated to walk through a greater number of rooms just because some edges disappeared. Also, for every action taken by him, a new noise is made and other edges and items are ignored, so the character needs to make new plans.

The PDDL problem description file is made with: a section that defines the domain of the problem, a section with the instances of the objects that compose the particular problem, and a section initializing these objects with a collection of predicates. In the system developed in this work, the information to be removed is chosen

randomly and to not be considered by the planner, it is just ignored as the problem description file is created. The removal of a node itself may be a challenging task depending on the problem to be solved. This operation is done by the removal of some predicates that compose the objects initialization, but only the ones with information that do not compromise the story basic structure. In the example above, it corresponds to leave in the world at least the items the character needs to collect and maintain the graph connected.

The use of an omission noise algorithm at first sight, appears to, at most, increase the size of the original story by disturbing the characters, but these noises also can make longer and/or more dangerous paths disappear helping the characters to reach their goals quickly.

4.2 Divergence Noise

The other type of noise to be considered in generating new possibilities for the development of stories is the divergence noise. As in the omission case, it may act in the characters' perceptions sensors or in the knowledge-base, but this time changing the information, once more forcing the characters to make new plans for each action they take.

In the example described above, the use of divergence noise could make changes in the edges of the graph or in the positions of the items. Different from the omission case which would still permit one character to make coherent plans, in other words, it still would be able to make a route using paths that do exist, with the divergence noise the character would try to use paths that lead to rooms that were not considered in the planning step or even use paths that do not exist. The errors in the position of the items would lead the character to wrong rooms too. As can be noticed, the use of divergence noise in the edges of a graph requires the management of the choices made by the character in his point-of-view and their real consequences.

While the omission noise may lead a character to have fewer action choices or make a correct, but not optimal plan because of removed information, the divergence noise may lead principally to wrong plans, because if a character is created being able to respond to every situation with a set of actions and if the information he uses to choose between actions is wrong, he necessarily makes wrong actions.

5 Tests

5.1 Scenario

For this work we used as test case the generation of a story inspired from one segment of Little Red Riding Hood. This act is performed by two characters: the protagonist and the wolf, and it takes place in a forest that the first character has to go through to reach her grandma's.

In our scenario, this forest is represented with a map, composed of paths that lead the protagonist to her goal. It has a set of nodes, each one with a name, a collection of neighbors and a danger value, which represents how problematic it is to pass through the place it represents. This danger value is used with one of the characters attributes, called tension, to monitor the level of excitement of the story development. Little red riding hood initiates the story in her house with the goal of reaching her grandma's with a bunch of flowers. The wolf begins the story with the objective of killing the other character. As the protagonist will not be able to walk after being killed, the goals are conflicting, so the story is programmed to end if any of the characters reach his own.

The available actions are: move between locations, with the effect of increasing the tension value of the executioner by the danger level of the destination, kill another character, and pick up object.

With the description given up until now, it is possible to generate different stories based in what paths the characters choose to take. This problem description alone would lead to variations based in the map and in the distribution of the flowers. To make different developments, the omission noise is inserted in the map description to remove some of the paths. This leads to a reduction of movement options, but considering that, in every turn different paths are removed, the plans the characters do in a first iteration may not be valid in a second one. To maintain possible the fulfilling of the protagonist goal, the omission noise is controlled to keep the map connected. The divergence noise is also used in the map description, but to change the locations of the flowers. The control over the divergence algorithm is used to allow the little red riding hood to pick the quantity of flowers it was assigned to. To evaluate the use of these two types of noise, the story generation process was done first without them, then with the insertion of only the omission noise, followed by the insertion of only the divergence noise and finally with both. For each case with noise, the system was executed 30 times.

5.2 Results

As a deterministic algorithm, the GraphPlan generated only one story for the test case that did not utilize any noise. This story is shown in code 2, it is composed of 6 actions, in which the wolf's and the Little Red Riding Hood's tension was raised to 5 units and the wolf ended successful.

```
[move(lrrh, house, woods)
 move(wolf, bridge, woods)
 move(lrrh, woods, church)
 move(wolf, woods, church)
 pick(lrrh, church, flower)
 kill(wolf, church, lrrh)]
```

Code 2: *Action sequence representing the story generated without the use of noise*

With the omission noise it was possible to obtain 10 different developments, in which 4 resulted in the death of the protagonist and 6 in the arrival at the grandma's. 24 story instances created those first 4 ends, and 6 instances the remaining 6 ends. In code 3, it is shown the generated story with higher tension development for this case.

```
[move(lrrh, house, woods)
 move(wolf, bridge, woods)
 move(lrrh, woods, bridge)
 move(wolf, woods, church)
 pick(lrrh, bridge, flower)
 move(wolf, church, bridge)
 move(lrrh, bridge, grandma-house)]
```

Code 3: *Action sequence representing one story generated with the use of omission noise*

For the third case, the generated process created 8 different developments, 5 of them ending with the victory of the wolf and 3 of them with the victory of the protagonist. 24 instances were responsible for the first 5 ends and 6 instances for the other 3 ends. The story created with only the divergence noise with higher tension development is shown in code 4.

```
[move(lrrh, house, woods)
 move(wolf, bridge, woods)
 move(lrrh, woods, church)
 move(wolf, woods, church)
 move(lrrh, church, woods)
 move(wolf, church, woods)
 pick(lrrh, woods, flower)
 kill(wolf, woods, lrrh)]
```

Code 4: *Action sequence representing one story generated with the use of divergence noise*

In the final test case, the system was able to find 15 different developments, 6 of them ending with the death of the Little Red Riding Hood and 9 endings with the fulfilling of her goal. 18 instances reached the first 6 endings and the 12 remaining instances the other 9 endings. This case was able to create an alternative story considerable higher than the others, this story can be seen in code 5.

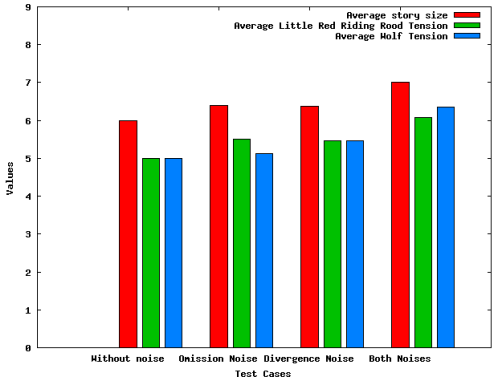

Figure 2: *Comparison of story size and characters' tensions.*

```
[move(lrrh, house, woods)
 move(wolf, bridge, church)
 move(lrrh, woods, marsh)
 move(wolf, church, woods)
 move(lrrh, marsh, bridge)
 move(wolf, woods, bridge)
 move(lrrh, bridge, marsh)
 move(wolf, bridge, woods)
 move(lrrh, marsh, bridge)
 move(wolf, woods, bridge)
 move(lrrh, bridge, church)
 move(wolf, bridge, church)
 pick(lrrh, church, flower)
 kill(wolf, church, lrrh)]
```

Code 5: *Action sequence representing one story generated with the use of both noises*

Tension development, in those test cases, is directly related to the path each character uses to complete his goals, in result it shows how in execution it was possible to make the character take different courses. In Figure 2 it is possible to see the average story sizes and average tension values.

6 Conclusion

The similarities in the final tension values for the two characters is a direct result of the behavior of the wolf and the map. While the protagonist has to reach a place with flowers and then proceed to the grandma house, the wolf needs to chase her (making him repeat her movements). Also, the size of the map and the wolf's initial position allow him to stay, at most, one node away from the other character. Moreover, this explains why the stories tend to end with the death of the protagonist (including the one without noises), because the wolf only has to chase her until she finds a field of flowers, and then while she uses her last turn to pick some, the wolf utilizes his last to attack.

The omission noise was able to aid the Little Red Riding Hood by removing some options that would lead the characters to an early encounter, so she had a safe distance to pick the flowers and proceed with her goals. In one story, this removal did the opposite effect: the protagonist moved directly to the wolf, whose only two actions were a move to a place without danger and the killing of the other character. The divergence noise, in some cases, put the flowers in the protagonist's house, which led her to pick them immediately, leaving the wolf with two options as initial movement and clearing the path to the gradma's house. In other cases, the flowers were put in the protagonist goal position, so it was possible to reach it and finish the story without giving chance to the wolf. In some stories the positions of the flowers changed in the same turn they were meant to be picked. Finally, with the two noises different situations emerged, like the positioning of flowers in nodes to which the characters "forgot" the path, sometimes combined with the removal of some path that would lead the wolf to the protagonist. In short, the use of more corrective actions led to stories with different sizes and tension levels.

The modeling of noises in the characters' visions of the world turned possible to generate interesting variations of stories for the same scenario. For our test cases it mostly increased the size of stories, but it also created smaller ones. Some combinations led to exceptional situations in which the tension of the characters suffered considerable variations. However it brought the necessity to treat all the new problems that a character may confront when it takes a wrong action, like dead-ends.

As a work in progress, we consider important to continue this research the use of noise with different planning algorithms to see how much impact it would cause in story development. Another direct evolution of this paper is the insertion of noises in the actions. This idea would repeat some proposes of this paper: increase the development possibilities, simulate characters that do not know exactly what are the consequences of their actions and make the generation process more realistic.

References

BOSSE, T., MEMON, Z., AND TREUR, J. 2007. Emergent storylines based on autonomous characters with mindreading capabilities. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IEEE Computer Society, 207–214.

BROVERMAN, C., AND CROFT, W. 1987. Reasoning about exceptions during plan execution monitoring. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-87)*.

CAVAZZA, M., CHARLES, F., AND MEAD, S. 2002. Character-based interactive storytelling. *IEEE Intelligent Systems,*, 17–24.

CHARLES, F., AND CAVAZZA, M. 2004. Exploring the scalability of character-based storytelling. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, IEEE Computer Society, 872–879.

DA SILVA, F. A. G. 2007. *Geração de Enredos com Planejamento Não-Determinístico em Storytelling para TV Interativa*. Master's thesis, Universidade Federal do Estado do Rio de Janeiro.

DAMIANO, R., AND LOMBARDO, V. 2009. Value-driven characters for storytelling and drama. *AI* IA 2009: Emergent Perspectives in Artificial Intelligence*, 436–445.

KLEIN, M., AND DELLAROCAS, C. 1999. Exception handling in agent systems. In *Proceedings of the third annual conference on Autonomous Agents*, ACM, 62–68.

KRUIZINGA, E. 2007. *Planning for character agents in automated storytelling*. Master's thesis, University of Twente.

MATEAS, M., AND STERN, A. 2005. Structuring content in the facade interactive drama architecture. *Proceedings of Artificial Intelligence and Interactive Digital Entertainment*, 93–98.

PORTEOUS, J., CAVAZZA, M., AND CHARLES, F. 2010. Narrative generation through characters' point of view. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, International Foundation for Autonomous Agents and Multiagent Systems, 1297–1304.

POZZER, C. 2005. *Um sistema para geracao, interacao e visualizacao 3D de historias para TV interativa*. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro.

RUSSELL, S., AND NORVIG, P. 2010. *Artificial intelligence: a modern approach*, 3 ed. Prentice hall.

SAIGOL, Z. 2007. Intelligent planning for autonomous underwater vehicles: Rsmg report 2. Tech. rep., School of Computer Science, University of Birmingham.

SU, W., PHAM, B., AND WARDHANI, A. 2007. Personality and emotion-based high-level control of affective story characters. *IEEE Transactions on Visualization and Computer Graphics*, 281–293.