Social-Network-Based Guided Emergent Narrative

Rose Abernathy
Haverford College
rose.abernathy@gmail.com

May 6, 2013

Abstract

Narrative is a key component of many video games; however, the interactivity of video games poses unique challenges to storytelling. An ideal video game narrative would react flexibly to the player's actions while still maintaining narrative integrity and following the author's intent. The goal of *narrative generation* is to balance flexibility and authorial intent without requiring every possibility to be planned for ahead of time. This paper describes previous work in narrative generation and proposes a new approach called Social-Network-Based Guided Emergent Narrative (SN-GEN). SN-GEN represents overall story plots as features of a social network graph. Individual agents in the story act autonomously, but are guided by the plot points. The game Sprite Quest is presented as an implementation of SN-GEN. Finally, SN-GEN is analyzed along two dimensions: expressive capability and flexibility to player interference.

Contents

1 Introduction

Video games are a unique medium because they combine the alternate worlds, characters, and scenarios of a film with the interactivity characteristic of real life. A video game narrative provides motivation and context for gameplay. Most modern games include some narrative elements, even when the story has little to do with the mechanics of gameplay. Narrative is often a major component of games; for example, the many games in the first-person-shooter genre are distinguished primarily by their setting, characters, and story.[1] Some games, called Interactive Narratives, are focused solely on the story and have no other gameplay.

Because video games are interactive, video game narratives must be different from the narratives of novels, films, and so on. The *Interactive Dilemma* is the conflict between the story that the author wants to tell and the freedom of the player to construct her own story [Peinado and Gervás, 2004]. If the game follows an authored narrative exactly, then the player cannot be allowed to interact with the story in case she gets in the way of the desired narrative.[2] Games with such linear narratives can still be appealing because on the quality of their story and their other gameplay merits, but they lack a fully interactive experience. The other extreme is a game without any narrative control.[3] The player may have total freedom to construct her own story, but there is no guarantee that the story will be interesting or dramatically sound. To fully take advantage of the medium, a video game should balance authorial control with user interactivity.

Of course, few games follow these extremes. Games with a strong narrative component often have branches, in which the player chooses which direction to take the narrative. A more complex form of a branching narrative is one in which the results of player's previous choices affect aspects of the story later on (for example, the player's relationship with other characters.) Another method of reducing linearity is sidequests, optional narrative or gameplay segments that do not affect the main narrative. These methods can be used to create stories that are engaging and interactive. However, they are limited by an *authoring wall*: the content for each branch or sidequest must be written ahead of time [Lucas et al., 2012]. As the player gains more and more freedom, the demands on the authors of the game get higher and higher. The content needed could double with each branch, even though most players will only experience one or two paths through the story. Therefore, the goal of video game narrative generation is to reconcile author control and interactivity without requiring all story content to be written by an author ahead of play time.

In this paper I present an approach to narrative generation in which the desired plot is represented as an attribute of a social network. With this representation, progress towards a story goal can be quantified as a function of the current state of the network. Non-player characters (NPCs) act autonomously, which means they can respond freely to any interaction from the player; however, their choice of actions is biased towards fulfilling the goal. The objective for this guided emergent narrative system is to give the player full

[1] One example is Mass Effect (BioWare, 2007), which is known for its complex and engaging science fiction plot and the ability for the player to develop relationships with various characters.

[2] A genre of games known as JRPGs (Japanese Role-Playing Games), including the Final Fantasy series (Square, 1987) is known for having totally linear storylines.

[3] These games, including The Sims (Electronic Arts, 2000), could also be described as simulations or sandboxes.

freedom of interaction while still allowing the author to direct the story.

The remainder of the paper is structured as follows: first a review of previous research in this area; then relevant background on social network analysis; next, a description of this project, the Social-Network-Based Guided Emergent Narrative System (SN-GEN); then details of the implementation of SN-GEN; finally, an evaluation SN-GEN.

2 Related Work

Narrative generation projects can be roughly categorized as bottom-up or top-down.

2.1 Bottom-Up Approach to Narrative Generation

The *bottom-up* or *emergent* narrative approach creates a story out of the autonomous actions of individual agents. This method is reminiscent of the real world, in which we perceive narrative structure emerging from the actions of individuals. (Aylett presents the example of a sports game, which can seem to have narrative structure even though each player simply acted in the way that would best serve the game [1999].) An emergent narrative allows for great player agency, since characters are free to simply react to the player's actions as they would to any other event. The downside to this approach is that there is no guarantee that an interesting or entertaining story will emerge, and that the author has no control over the story except for setting the initial state. Projects that use a bottom up-approach include Virtual Teletubbies, The Prom, and the Continual Multiagent Planner [Aylett, 1999; McCoy et al., 2010; Brenner, 2010].

2.1.1 Virtual Teletubbies

Aylett identifies three levels of a narrative: the overall plot, character actions, and atomic behaviors [1999]. In the most scripted case, an author controls all three levels. Another possibility is for that an author decides the overall plot, but agents "improvise" their own actions; this is a top-down approach. According to Aylett, improvising at the overall plot level can only be done by only human authors.[4] Rather than require creativity from a machine, this paper describes a bottom-up method where the top level of the narrative is determined by the lower levels.

Aylett presents an emergent narrative system called Virtual Teletubbies [1999]. The Teletubbies are motivated by drives such as hunger and curiosity, which prompt them to pursue certain goals. Although this simulation could form an interesting narrative, the author correctly points out that this is not guaranteed. Without any authorial input, there is no mechanism for focusing on important narrative events and skipping uninteresting interludes.

The player can explore the virtual Teletubby environment, but not interact with it. Aylett is concerned about how to ensure that a player would go along with the emerging narrative if player interactions were allowed. One solution would be to include a "social presence" in the game that pressures the player to follow along. For example, a player who cheats in a sports game will be booed by fans. Although this is an interesting consideration,

[4]There are recent projects which dynamically generate plot events, such as PaSSAGE [Thue et al., 2010].

it is not clear how the system would determine what is acceptable or unacceptable behavior. One of the key features of an emergent system is its ability to incorporate player actions *into* the story instead of working around them.

2.1.2 The Prom

"The Prom" is an emergent narrative game which includes elaborate social relationships between players [McCoy et al., 2010]. The Prom is built on Comme il Faut (CiF), an AI system designed to support relationships and social interactions. McCoy et al. choose a high school setting that would highlight characters' complicated social lives.

Figure 1: Screenshot from The Prom [McCoy et al., 2010]

CiF models social interactions as "social games" with preconditions, initiator influence that determines which of all possible actions get priority, responder influence that determines the result of the interaction based on factors such as the current relationship between the agents involved, and possible effects. The characters are connected via a social network with three types of reciprocal links: dating, friends, and enemies. Additionally, each character has an opinion about others in terms of three features: *buddy* (amiability), *romance*, and *cool* (respect). Every character also has a cultural knowledgebase, including facts and personal values, as well as a social facts database that remembers previous social interactions.

The authors hope that a varied and entertaining story will emerge when all these aspects interact. For example, two characters might become friends because they like the same things, then start dating; but if one was already dating someone else, he would become known as a cheater; and so on. The role of the player in The Prom is quite interesting. Rather than representing a single character, the player can control any character; which actions are available to that character and how other characters respond are still determined by the model. Additionally, the player is given goals such as "Get John a date to the prom." Because of this, the resulting story is authored more by the player than by an emerging narrative. But regardless of this implementation, it is clear that the CiF system could also be used for a game with truly autonomous agents. CiF creates complex agents that have a

lot of potential for interesting interactions, which are the building blocks of an interesting story.

2.1.3 Continual Multiagent Planning

For NPCs to be believable as characters, they must be goal-directed. The virtual Teletubbies had simple goals relating to their basic drives, and the characters in The Prom worked towards the player's goals. Characters using Continual Multiagent Planning (CMP), can create complex multi-agent plans to accomplish their goals [Brenner, 2010].

Brenner makes use of the Multiagent Planning Language (MAPL) to formally describe the actions that make up a plan. Unlike in other planners, actions in MAPL include knowledge preconditions (not only must the conditions be right for this action, the controlling agent must know that the conditions are right) and commitment preconditions (whether other agents have agreed to perform the actions necessary for this plan.) CMP is a distributed planning algorithm that generates MAPL plot graphs for the characters. A player can participate by choosing MAPL actions directly.

The author has applied CMP to a "quest" domain in which the king wants to get his treasure back from a dragon. To do so, he must enlist the help of other characters. Although the author can provide characters with initial goals, the actual plot is decided by individual agents' actions. CMP is able to make dynamic plots, in which characters' beliefs and personality traits may change. However, like other bottom-up approaches, there is no guarantee of plot quality. Possible solutions presented by the author include adding an *author agent* that tries to achieve plot goals by enlisting other characters, or *reader agents* who asses the quality of the plot as it unfolds.

Bottom-up narrative generation is characterized by a "hands-off" approach. The author controls the initial state (which can include characters' social history in the case of The Prom [McCoy et al., 2010], or story-related goals as in CMP [Brenner, 2010]), but has no input once the story starts. Additionally, the player does not necessarily have a central role in the narrative. In fact, the story could proceed fine without any input from the player, as in Virtual Teletubbies [Aylett, 1999]. These factors make it difficult to ensure that the resulting narrative has the qualities that the author desires. However, they provide a big advantage over a top-down set-up in the low authoring cost (setting up the initial state once could generate a huge number of distinct stories) and the agency of the player, who can perform any action the world allows without interfering with a designated plotline.

2.2 Top-Down Approach to Narrative Generation

In a *top-down approach*, an author describes the storyline and the narrative system fills in the details. As the story progresses, the system reacts to the player's actions while maintaining the integrity of the narrative. Because of the greater involvement of a human author, this method is more likely to produce a "good" narrative. But since the player's actions might derail the intended story, the system must either work around player interference or restrict player freedom. A common way of implementing the top-down approach is with an *experience manager* which directs NPCs to follow the story and tries to correct for player interference [Riedl et al., 2011].

2.2.1 Haunt 2

The goal of the top-down approach is to present a structured narrative which still allows for player agency. Haunt 2 is an interactive drama in which a *director* (sometimes called an experience manager or drama manager) ensures that the script is followed no matter what the player does [Magerko et al., 2004]. In Haunt 2, the player takes the role of a ghost who must find out who killed her and bring that person to justice. The player's interactions are limited to scaring NPCs by appearing or disappearing, and possessing NPCs to influence their decisions. Even with only these two powers, the player has a lot of freedom to explore and interact with different characters at different times.

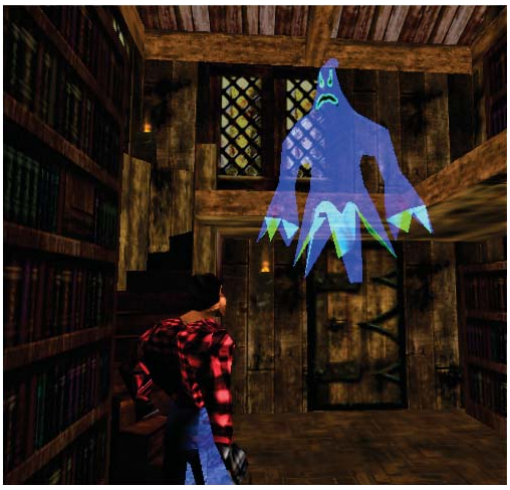

Figure 2: Screenshot from Haunt [Magerko et al., 2004]

NPCs in Haunt 2 are semi-autonomous. If nothing else is happening, they pursue their own goals, such as visiting the room with a fireplace when they are cold. However, the director has the ability to issue directions to the NPCs. The directions can be high level, such as telling a character to act sociably, or low level, such as providing the exact lines of dialogue needed for a scene.

By giving instructions, the director makes sure that the player experiences the author's plot points. For example, the author wants the player to overhear a conversation at the beginning of the game. The precondition to this plot point is that the player and the two NPCs involved are in the same room. If necessary, the director interferes and tells the NPCs to move to the player's room so that the precondition will be satisfied within a given timeframe and the plot can proceed. The director also makes predictions about the player. As the player progresses, the director updates a model of what the player knows. For example, for the player to get an NPC to discover the dead body, she must first know there is a body. If the model fails this precondition, the director steps in.

This experience manager system combines autonomous agents, apparent player freedom, and author control. But although the system accommodates various player actions, the player does not have much impact on the story itself; instead, the system tries to ensure that the original plot proceeds. The player's limited impact on the story is underscored by her limited methods of engagement.

2.2.2 Automated Story Director

The Automated Story Director (ASD) created by Riedl et al. can respond to player interference in a more flexible way [2008]. Narratives are represented by partial-order causal link (POCL) plans. A POCL plan is a directed acyclic graph of actions which affect the world state. A causal link exists between two actions if the first action causes condition c to be true, and c is a precondition for the second action. For example, getting a key might be causally linked to unlocked a door. Actions can also be connected by temporal links, which impose the constraint that one action must occur after another. The POCL plan is searched by a partial-order planner (POP). POP is a refinement search algorithm which inspects a given plan, finds flaws, and attempts to revise the plan to correct those flaws.

The ASD takes as input an *exemplar plan*, which is the author's preferred story, and follows that plan as closely as possible. A player's interactions in the game are characterized as *constituent* with the exemplar narrative if they go along with it, *consistent* if they have no effect on the narrative, or *exceptional* if they change the world so that the story cannot continue as intended. If the action is exceptional, the ASD must find the next best trajectory by removing actions that are now impossible and then invoking the planner to fix flaws. In order to control the narrative more closely, the author can also choose *islands*, or states through which all solutions much pass. If it ever becomes impossible to re-plan a valid story, the system defaults to an emergent state where characters act on their own.

Once the ASD has a plan, it directs the characters by giving prescriptive directives (to instruct the character to advance the narrative) or proscriptive directives (to prevent the character from violating any preconditions necessary for the narrative to continue). The characters are semi-autonomous and can handle basic player interaction themselves.

The process of planning around possible exceptional player actions is done offline, before the game begins. This results in a tree of plans, where each branch diverges further from the exemplar narrative. During the game, the ASB switches to the correct plan in reaction to player actions; this avoids the delay of re-planning in the middle of the game. This enumeration of all possible player interference could be quite extensive depending on the size of the narrative's state space and the amount that the player can interfere. Even though the result is a branching storyline that resembles a fully-authored narrative, the author only has to provide one exemplar plot.

2.2.3 Façade

Façade is an interactive narrative game with a focus on believable characters and joint behaviors [Mateas and Stern, 2003]. In the game, the player visits Grace and Trip, a couple whose marriage is falling apart. It is noteworthy that the player can interact freely by typing whatever she wants to say, although the meaning of complicated statements does not always come across.

Characters in Façade are programmed in ABL (A Behavior Language) [Mateas and Stern, 2002]. An ABL program is a collection of behaviors. Sequential behaviors include a sequence of steps; if a step fails, the whole behavior fails. Joint behaviors ensure that the behavior of two characters who are meant to be interacting or working together line up. Characters also have short-term memory, which contains working memory elements (WMEs.) For example, when someone knocks, a KnockWME is placed in the agent's mem-

ory. The narrative is structured around *beats*, or atomic story components. For example, the "answering the door" beat requires Trip to open the door, greet the player, and invite her in. An experience manager adds or removes possible beats depending on the situation.

The result is a fluid series of events that reveal more about the NPCs and can result in different endings to the narrative. In some ways the system is bottom-up, since beats are chosen as the game continues and the story can vary greatly. However, the story comes from combinations of authored events rather than emerging from agent interactions, so Facade has more in common with top-down systems. A weakness of ABL is the difficulty of accommodating player interaction; almost every behavior requires its own rules for how to handle player interruptions. Systems with more limited agent behaviors or player actions require less authoring for the initial set-up.

The experience manager approach used in these projects start with an authored narrative and make it more flexible to allow for player interaction. These systems could potentially replace typical video game narratives without losing the well-structured plot. However, player agency must still be managed to prevent the player from derailing the plot. In the ASD project, it is possible to reach a world state where the goal cannot be achieved [Riedl et al., 2008]; this is not acceptable for the narrative of a full video game. On the other hand, Haunt 2 and Façade limit the player's interactions to a form that cannot interfere too badly with the plot (haunting and conversing respectively) [Magerko et al., 2004; Mateas and Stern, 2003]. Additionally, crafting complex behaviors and plans like those in Facade requires a lot of authoring, and one of the goals of narrative generation is to avoid relying too much on human authors.

3 SN-GEN

3.1 Description

Each of the projects presented above strikes its own balance between authorial control and player agency. I would like to propose a interactive narrative system that has the flexibility and low authoring requirement of an emergent narrative, but can still be guided to follow a target plot. This idea is called the *guided emergent narrative*, or GEN.

A key component of an emergent narrative system is that the individual agents are autonomous and follow their own goals. These initial goals may be carefully crafted by the author, chosen randomly, or determined completely by the story so far. In GEN, the agents assess progress towards their goal using a heuristic that evaluates the current game state. Since player actions are reflected in the game state, agents can seamlessly react to player interference without any special mechanism. This is an important asset of GEN and other emergent narrative generators: since the narrative is composed of the actions of individual agents, there is no way for the player to "break" the narrative.

In a purely emergent narrative system, the author controls the initial set-up, but not the progression of the story. The stories generated this way might not follow the plot that the author wished for. Ideally the author could guide the stories as it progresses to ensure that it follows a satisfying narrative arc and perhaps to aim for a certain ending. In GEN, agents are guided by the author to meet certain plot points, while still maintaining their

autonomy. The plot points are incorporated via a global heuristic that influences agents' decisions.

When choosing the next action to take, an agent evaluates the result of each action using both his local heuristic and the global heuristic. For example, an agent may choose to take an action that does not advance his personal goal if it greatly advances the plot point. On the other hand, it's unlikely that he would take an action that is completely "out-of-character" (works against his goal) because the value of that action by his local heuristic would be very low. Additionally, the weight of the global heuristic can be adjusted. If the global heuristic has a low weight, the story will resemble a pure emergent narrative in which each agent follows his own goals; if the global heuristic has a high weight, the story will follow the plot points more directly. This weight could be changed during gameplay; for example, if the player's actions lead the story away from the goal, the weight could be increased in order to get the story back on track.

GEN could be implemented with various heuristics and representations of the game state. In this paper, I have chosen to focus on a social network representation. GEN is able to deal with multiple agents whose interactions affect one another, so social interactions are a good behavior to focus on. The plot points determined by the author are also phrased in terms of social network analysis. For example, the plot point "All the NPCs have split into two warring factions" could be represented by a graph with two distinct clusters. The rest of this paper will focus on the social-network-based version of GEN, or *SN-GEN*.

3.2 Formalization

Let $A = \{a_0, a_1, a_2, ..., a_n\}$ be the set of n agents, and let S represent the current game state. There are a defined set of actions which modify the game state in some way; i.e. $f_i(S) \rightarrow S'$. An action may have a precondition $p_i(S, a)$, where a is the agent that performs the action, such that actions may be unavailable due to the current game state or features of the agent.

Every agent a has a local objective function, $o_a(S)$, which returns a real number between -1 and 1. 1 indicates that the state is very desirable to the agent, and -1 indicates that the state is very undesirable. In addition, there is a global objective function $o_{global}(S)$, which similarly returns a number between -1 and 1. There is also a global weight, w, which indicates the weight each agent assigns to the global objective function versus their local objective function.

An agent chooses a new action by looking at every possible action and picking the one which performs best on a weighted combination of the objective functions. The value of an action i to an agent a is given by a weighted combination of the objective functions:

$$v_{a,i}(S) = w * o_{global}(f_i(S)) + (1 - w) * o_a(f_i(S))$$

An agent will prefer to choose the action with the highest value ($argmax_i\ v_{a,i}(S)$), although some randomness may be incorporated into the choice of action to encourage more dynamic situations. Once an action is chosen, it may be applied immediately or be added as the agent's current goal: the narrative will be more evident if it involves the agents working towards their goals rather than achieving them instantly. A new action may be chosen when the current goal is complete, whenever the game state changes, or on some other trigger.

Due to the potentially large number of possibilities that must be considered, it is wise to reduce the frequency with which agents choose new actions to avoid delaying gameplay.

In the social network version of GEN, N is a network, represented as an $n \times n$ matrix. $N_{i,j} \in [-1, 1]$ is the attitude of agent a_i towards agent a_j. N may begin as a 0 matrix or be assigned a start configuration. Every $N_{i,j}$ can be modified by actions during the course of the game except for $N_{i,i}$ which is always 0. In SN-GEN, the network contains all the game state information ($S = N$) and actions only affect the network.

3.3 Social network analysis

Social network analysis (SNA) is a technique for representing and analyzing relationships between people or other agents using a graph. In a social network graph, the nodes are the actors in the network, and the edges represent relationships between the actors. In its most basic form, an undirected edge indicates the presence of a connection between two agents. Edges can also be weighted to indicate the strength or mode of the relationship. Typically, larger weight indicate a stronger relationship. Positive weights can be used to indicated positive relationships (e.g. friendship) and negative weights to indicate negative relationships (e.g. conflict). Additionally, a social network graph can be directed; in that case, an edge from agent A to agent B indicates A's attitude towards B, which might differ from B's attitude towards A. [Krebs, 2011]

SN-GEN uses a network in which each node represents one NPC in the game. For simplicity, everyone node is connected to every other node; there are no people who do not know each other. The edges are directed and have weights between -1 and 1. -1 means hate, 0.5 means like somewhat, 0 means neutral, and so on. Because of these properties, SN-GEN uses a modified version of some basic network attributes:

3.3.1 Edges

One of the simplest attributes that might be used in a narrative is the relationship between two characters. Let a and b be agents; then we can easily find a's attitude towards b by looking at the weight of the edge from a to b: $attitude(a, b) = weight(a, b)$. Additionally, it might be useful to know the mutual relationship between two characters: $friendship(a, b) = (weight(a, b) + weight(b, a))/2$.

3.3.2 Indegree

Indegree measures popularity, or the amount an agent is liked by others. Indegree typically refers to the number of edges entering a node, but because of the weighted edges in SN-GEN, we want to consider the *sum* of the edges instead. Let a and b be agents, and A be the set of all agents. Then indegree can be calculated by: $inDeg(a) = \sum_{b \in A} weight(b, a)$. A high indegree indicates high popularity, and vice-versa.

3.3.3 Outdegree

Outdegree measures an agent's attitude towards others, based on the sum of the edges leaving that node. For SN-GEN, outdegree is calculated as: $outDeg(a) = \sum_{b \in A} weight(a, b)$. A

high outdegree corresponds to an agent who likes most other characters, and a low outdegree corresponds to a agent who dislikes most others.

3.3.4 Cliques

A clique is a group of nodes in which every node has an edge to everyone other node in the graph. For SN-GEN, we will consider a node to be in a clique if it has a positive edge to and from every other node in the clique. Therefore, a clique represents a group of people who are all friends. This concept could be very useful for telling stories with SN-GEN; unfortunately, there is no efficient algorithm for finding cliques. Still, it is possible to implement cliques via an enumerate-and-test algorithm. Although cliques can be as small as two nodes, we will define a clique as significant if its size is greater than 1/5 the number of agents. We will limit the uses of cliques to two cases, which are more appropriate as global goals than local goals:

1. Deciding if agents a and b are both in the same significant clique.
2. Deciding if agents a and b are each in a different significant clique.

3.3.5 Density

The density of a network is the proportion of edges that exist compared to the total number possible. Since SN-GEN uses positively- and negatively-weighted edges, summing the weights of all the edges will give us a measure which indicates the overall friendliness in the network. This can be calculated as: $friendliness(A) = \sum_{a,b \in A} weight(a, b)$.

3.4 Possible Uses of SN-GEN

SN-GEN takes plot points (which could be specified by a human author or a higher level of narrative generator) and flexibly fills in the details with the actions of individual agents. It could be used in an interactive narrative game in which the player is free to interact with the story as she pleases, but the author ensures that she experiences a dramatic narrative arc. For example, in an interactive version of "Romeo and Juliet", the player could choose any two characters as the main couple. SN-GEN could then fill in the details of how the families of the lovers became enemies, how the two fell in love, and so on. Using SN-GEN, the author could set some details of the story and leave others to be determined during the game depending on the player's actions.

Games with action beyond that of an interactive narrative could also use SN-GEN to create dynamic background scenarios. Whether the player experiences the narrative directly or learns about it later, the complete series of events leading up to the current plot point will preserve the illusion of an organic story rather than one directed with a heavy hand. For example, a complex and open-ended RPG such as Skyrim[5] contains many towns that the player revisits during the course of her adventures. The player often witnesses discussions and conflicts between characters, which eventually lead to sidequests. In Skyrim, human authors wrote each of the discussions and planned out the series of events in each sidequest

[5](Bethesda, 2011)

by hand. With GEN, the authoring burden could be reduced while still providing complex and dynamic storylines for the player to explore. For example, if the author plans a storyline in which a certain NPC rises to power, becomes corrupt, and must be deposed by the player, the details of the NPC's rise to power could be generated on the fly by GEN. This might interest a player who begins to notice changes in the influence structure of the town; even a player who is not so curious would probably appreciate noticing some "foreshadowing" more than seeing an NPC rise to power out of the blue.

In general, SN-GEN could be used to generate details of any social relationships between NPCs. An appealing feature of the farming simulation games in the Harvest Moon series[6] is getting to know the NPCs. The player can even marry and start a family with certain characters, if they become close enough; however, if the player does not act quickly enough, a romantic rival marries the character instead. SN-GEN would be ideal for filling in details of town life, including the progression of the rival romance.

4 Implementation: Sprite Quest

4.1 Goals

For the purposes of this paper, I wanted to create a game that implements SN-GEN. This game should demonstrate how SN-GEN might work in the context of a more complete game, and especially how it deals with player interference. To provide such a demonstration, the game must do the following:

1. Provide a stage for social interactions between NPCs.
2. Present the interactions and their results to the player.
3. Allow and encourage the player to interact in ways that advance or interfere with the story goal.

Informing the player of the story goal directly does not seem useful; encouraging the player to work towards the goal would not show SN-GEN's flexibility, and encouraging the player to work against the goal is likely to end in frustration, since SN-GEN is expected to eventually reach its goal. Therefore, the game should also:

4. Provide the player with a quest that keeps her involved with the narrative but does not indicate the author's goals directly.

The game should also be playable, ideally with intuitive controls, appealing graphics, and an useful user interface. However, it is intended primarily as a demonstration of SN-GEN; the creation of a complete game with complex and fun gameplay is beyond the scope of this work.

[6](Marvelous Interactive, 1996)

4.2 Gameplay

The game I created to demonstrate SN-GEN is called Sprite Quest. The player's character in the game is a Sprite, a supernatural creature who enjoys interfering with the lives of humans. There are two possible quests the player can pursue: to become a Brownie, she must help a lot of humans (and work with the story goal); to become an Imp, she must play tricks on a lot of humans (and work against the story goal). These two quests will demonstrate GEN's resilience to both positive and negative interference.

(a) The player receiving a quest from an NPC (b) The player completing a quest

Figure 3: Screenshots of Sprite Quest

The game features only a few basic mechanics. The player can walk around the town and talk to the NPCs in it. Characters give her quests that potentially progress towards the story goal; for example, asking her to pass their compliments on to another NPCs, which would raise the friendship level between the two. The player then has the option to either fulfill the quest as specified, increasing her Brownie points, or to do the opposite of the quest (such as passing on an insult rather than a compliment), which increases her Imp points. The game ends when the player gathers the set amount of points for either the Brownie or Imp quest. Additionally, NPCs can interact directly with NPCs, such as by chatting together to increase their friendship. The player can observe these interaction, hear gossip from NPCs, or view a list of recent interactions. The story goals are described in an XML file read by the game. Although editing the story goals will not change the core gameplay, it will change the trends in the NPC's interactions. A message is displayed when each goal is reached to clue the player into the story and indicate the progress of SN-GEN.

4.3 Implementation

4.3.1 General

Sprite Quest was implemented as an Adobe Flash game, with the code written in Actionscript 3. Actionscript 3 is an object-oriented programming language targeted for Flash development. Additionally, an open-source 2-D game engine called Flixel (which provides the implementation of 2D graphics and spritesheets, basic physics and collisions, game

states and user input, among other features) was used as the framework for the game. This platform was chosen because I have previous experience with it, and because Flash games can be played online and shared easily. The graphics used in the game were either created for this game, or reused from a previous projects. No sound or music is used in the game.

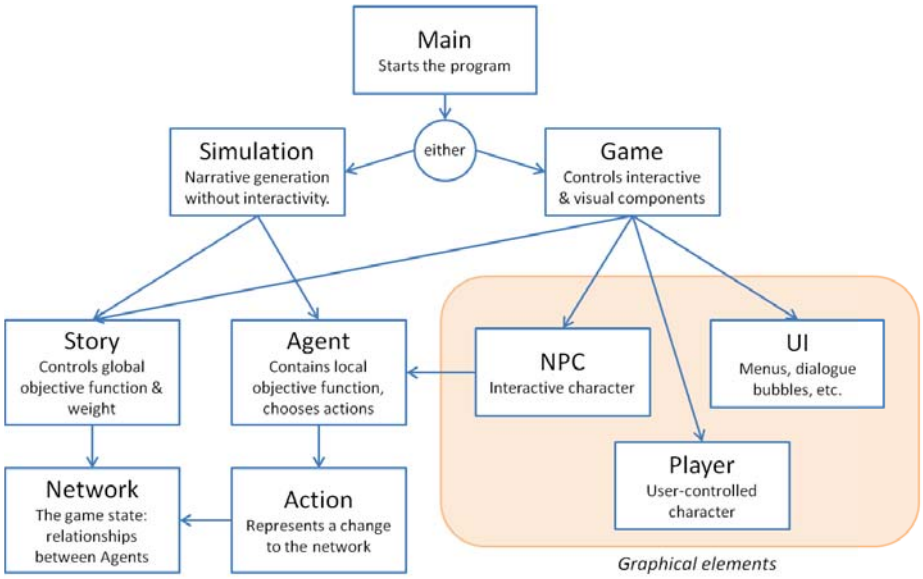

Figure 4: Organizations of the main classes in Sprite Quest. Arrows indicate has-a relationships.

The diagram in Figure 4 shows the structure of the classes used in the game. The program can be started in either the Simulation or Game mode. The back-end narrative generation system, including the collection of individual Agents and a Network representing the game state, can be run automatically from the Simulation mode. Via the Simulation, it is possible to test the narrative generation system without playing through an interactive game every time. The Game mode, on the other hand, provides an interactive, graphical front-end interface for the narrative generation system. The behavior of each NPC in the game is based on the actions chosen by an underlying Agent.

An important feature of the Game implementation is the interaction between the Player and the Agents. To manage Actions in an interactive way, the NPCs implement a state machine which covers the different stages of choosing and completing an Action. See Figure 5 (page 16) for a diagram of the state machine. An Action object represents a possible future change to the game state; each NPC can have a single Action that it would most like to complete (i.e. that returns the highest heuristic value) as its *intent*. Agents currently in the Idle or Intent state have a chance to re-select an intent whenever any other Agent completes an intent. There is no need for Agents to constantly re-evaluate their intents because the game state only changes when an intent is completed.

There are two ways for an Agent to complete his intent: either by finding the target of the intent himself or having the intent completed for him by the Player. For example, suppose Agent Brian intends to complete the Action $<B$ *argues with* $A>$. He will begin moving towards his target, Agent Alice. However, the Player can intercept him and begin a

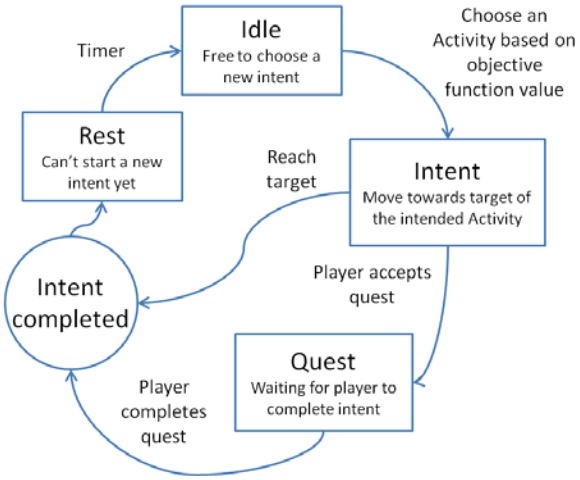

Figure 5: The state machine used by NPCs to choose and complete intended actions.

conversation during which Brian asks for the Player's help. If the Player accepts the quest, Brian enters the Quest state and no longer tries to reach Alice on his own. When either Brian or the Player complete the request, Brian enters a brief Rest state before choosing a new intent.

For the playable game and its complete source code, please contact this paper's author.

4.3.2 Objective Functions

The following tables name each objective function implemented in Sprite Quest and its corresponding social network concept. Let a be the agent in the local objection function, and let a, b be the targets of the story goals. Please refer to page 11 for additional definitions.

Agent Goal	Objective Function	Description
Friends	$\frac{1}{n-1} \times \lvert\{b \in A \mid friendship(a,b) > 0\}\rvert$	Have many positive / negative relationships.
Enemies	$\frac{1}{n-1} \times \lvert\{b \in A \mid friendship(a,b) < 0\}\rvert$	
Liked	$\frac{1}{n-1} \times inDeg(a)$	Be liked / disliked by other agents.
Disliked	$\frac{1}{n-1} \times -inDeg(a)$	
Neutral	$1 - \frac{1}{n-1} \times \Sigma_{b \in A} \lvert friendship(a,b)\rvert$	Keep relationships close to neutral.

Story Goal	Targets	Objective Function	Description
Popular	1	$\frac{1}{n-1} \times inDeg(a)$	A is liked/disliked.
Unpopular	1	$\frac{1}{n-1} \times -inDeg(a)$	
Friends	2	$friendship(a, b)$	A and B are friends/enemies.
Enemies	2	$-friendship(a, b)$	
Positive	0	$\frac{1}{n} \times \Sigma_{a \in A} inDeg(a)$	Most people have positive / negative relationships.
Negative	0	$\frac{1}{n} \times -\Sigma_{a \in A} inDeg(a)$	
SameClique	2	$\frac{1}{n} \times maxCliqueWith(a, b)$	A and B are both in a large clique.
DifferentClique	2	$\frac{1}{n} \times maxDifferentCliquesWith(a, b)$	A and B are in two distinct large cliques.

Except for the clique functions, each objective function strictly increases with progress towards the goal, which allow SN-GEN to achieve its goals without any planning. *SameClique* and *DifferentClique* are a slightly different case. Since they merely count the max cliques, the objective function does not reward actions that may lead to a larger clique. For example, if Alice and Brian are friends and Carl is not, Carl could create a larger clique by becoming friends with both Alice and Brian. However, becoming friends with just one of them, which is a necessary first step, would not be rewarded by the *SameClique* objective function. Future work on the subject could improve the objective function or incorporate some amount of planning. In the current implementation, the clique functions are still useful as long as there are enough agents whose personal goals prompt them to make friends. An example of this situation is found in the next section.

4.4 Results

4.4.1 Data

The narrative generated by SN-GEN in Sprite Quest is affected by several variables. The global objective function, or story goal, is set by the author. The number of agents can also be set. For every agent, the author can specify an local goals, or allow the agent to select a goal randomly. Additionally, the author can control the balance between the global and local goals via a parameter called *target steps*. Target steps indicates the approximate number of actions the author wishes to occur before the story goal is met. If the number of target steps is lower, the weight of the global goal versus the local objective functions will be increased more quickly so that the narrative is more likely to resolve sooner.

The relationships between some of these parameters was tested in the Simulation (non-interactive) mode. The results of the first test, which examined the effect of the target steps variable, are shown in Figure 6. Although the value of target steps does not correspond

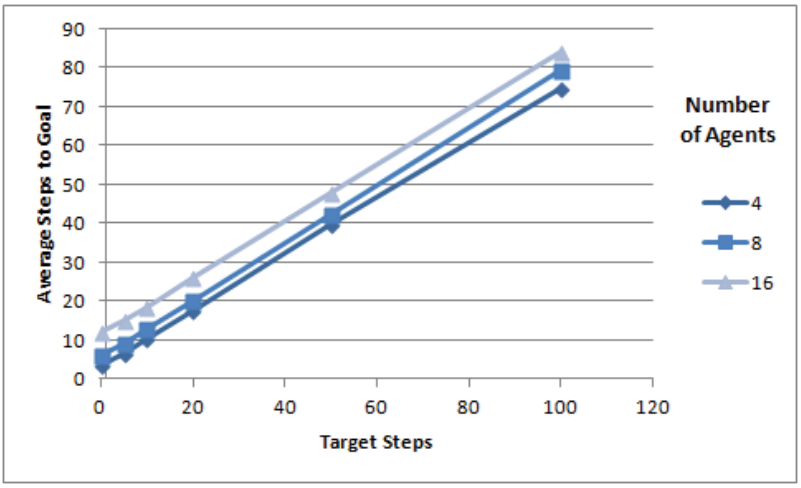

Figure 6: The average number of actions required to achieve the story goal over 50 trials, compared to the Target Steps parameter. In this test, the story goal is Popularity and the goal of all agents is Neutral.

exactly with the actual steps to goal, the two are clearly correlated. These results show that the author can control how quickly the story goal is reached. It may be possible to tweak the implementation of target steps so that the actual steps needed match even more precisely.

Another topic of interest is the relationship between the global objective function and individual agent objective functions. Figure 7 shows some of these relationships. The data indicates that a positive story goal is achieved more quickly when agents also have positive goals. The author may want to set each agent's local goal in order to tell a certain story, but it is also important to consider how local goals effect the number of steps necessary to complete the story. This example also demonstrates how ineffective clique goals can be when agents do not "cooperate".

4.4.2 Example

The SN-GEN implementation was tested with a longer narrative made up of several story goals. I chose the example of Romeo and Juliet, with agents A and B (a.k.a. Alice and Brian) as the star-crossed lovers.

Step 1: Feud The narrative begins with 8 agents without any opinions about each other. The first step of the narrative is to divide the agents into two opponent factions like the Montagues and Capulets. The story goal *DifferentCliques* is used, with Alice and Brian as the two inputs. The individual agent goals are assigned randomly except for the protagonists Alice and Brain, who are given the *Friends* goal.

After only 8 steps in which agents make friends, two families are established: Alice, Eric, and Gina form one clique and Brian, Denise, and Holly form another. Due to the limitations of the *DifferentCliques* function, the two families are not enemies, but merely ambivalent about one another.

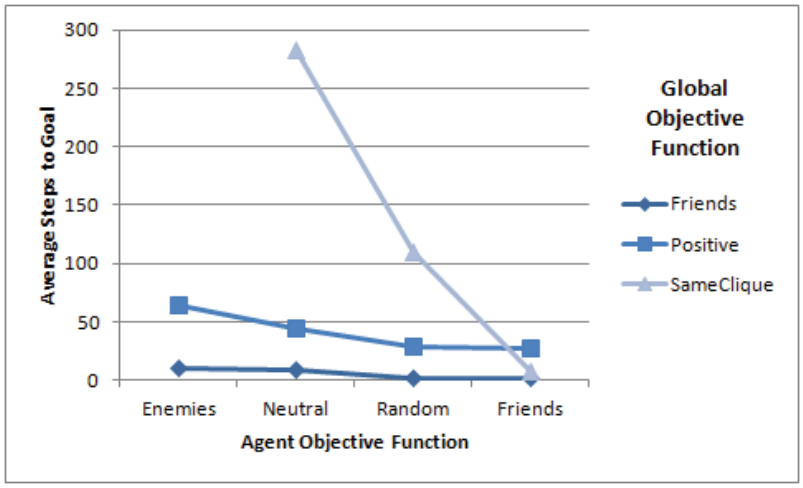

Figure 7: The average number of actions required to achieve the story goal over 50 trials depending on the local and global objective functions. The target steps number was 20 for Friends and Positive, but 10 for SameClique due to its slower nature.

Step 2: Love The next step is to make Alice and Brian fall in love, for which the *Friends* goal is a good choice. After 6 steps, Alice and Brian have established a strong rapport (indicated by an attitude of 1.0 towards each other.)

Step 3: Reconciliation Luckily for Alice and Brian, SN-GEN does not have any representation of character death. Instead, this phase of the narrative focuses on the reconciliation of the two families by requiring a high amount of *Positive* relationships. Carl, whose goal is to make as many *Enemies* as possible, threatens to ruin the peace by arguing with Denise; but after 18 steps Carl, Denise, and almost everyone else are friends.

In this example, SN-GEN successfully filled in details about the story goals, such as exactly what order people joined up into cliques. Unfortunately, a fairly limited set of possible goals prevented a dramatic story with a true family feud from taking place. The extent to which this implementation of SN-GEN can express various narratives is discussed in the *Expressiveness* section below.

5 Evaluation

5.1 Expressiveness

SN-GEN has an advantage over a pure emergent narrative because it incorporates guidance from an author. But what use is guidance if the author cannot tell the story he wants to tell? One axis along which we can evaluate narrative generation systems is *expressiveness*. A system that is able to express a larger variety of stories is better because it can be used in more scenarios and more types of games. In order to compare systems objectively, we must quantify the variety of stories that each can express. I propose referring to Friedman's

(supposedly) comprehensive list of 14 story plots [1955]. Whether or not this list is correct or complete is outside the scope of this paper; it covers a large variety of possible plots and will suffice as a method of evaluating a narrative generation systems.

In the following table, I examine each of the 14 story plots and give the narrative system 2 points for being able to express the scenario completely, 1 point for being able to partially express the scenario, and 0 points if the scenario cannot be expressed. According to this rubric, a fully expression narrative system will receive a total of 28 points. The score that SN-GEN receives will indicate the variety of stories that it can generate.

Plot type	Description	score	Example of SN-GEN plot
Action plot	Action-based, sequential story.	0	
Pathetic plot	A weak character suffers.	2	A character is shunned and disliked by everyone.
Tragic plot	A strong character fails.	1	A character who was admired becomes disliked.
Punitive plot	A bad character gets their just deserts.	1	A character who acts mean towards everyone else becomes disliked.
Sentimental plot	A weak character succeeds.	1	A character without many strong connections suddenly becomes popular.
Admiration plot	An ordinary character succeeds through integrity.	2	A normal character gains influence and becomes a leader.
Maturing plot	A character matures (often, coming of age).	0	
Reform plot	A fallen character restores their position.	2	A character who used to be influential wins others back to his side.
Testing plot	The integrity of a noble character is tested.	0	
Degeneration plot	A good character falls into immorality or disgrace.	1	A character who disliked a “bad“ clique eventually joins them.
Education plot	A character becomes a better person due to experience.	0	
Revelation plot	A character comes to accept a hard truth.	0	

Plot type	Description	score	Example of SN-GEN plot
Affective plot	A character's feelings change, often despite their rational consideration.	1	Two characters fall in love, despite belonging to distinct cliques.
Disillusionment plot	A character abandons their high ideals.	0	

SN-GEN receives 11/28 points in this analysis. Although it is able to create stories that deal with a variety of social situations, it lacks the capability to express non-social situations (such as action) and character's individual feelings. In fact, this method of evaluation may even be overly generous: scenarios such as action are even more common in video games than novels, which were the basis of this list. Because of its limited expressiveness, SN-GEN may only be useful in certain kinds of situation, such as those described in the "Possible Uses" section. In the next section, we will see how well SN-GEN can perform in such a situation.

5.2 Flexibility

The distinguishing feature of all non-linear video game narratives is their ability to respond to the player's choices. On the *flexibility* axis, an ideal narrative generation system would preserve the author's intent while still allowing a great amount of interference from the player. In other words, flexibility is the ability of the narrative to "bounce back" to a story consistent with the author's goals, no matter what the player does. The possibilities for player interaction vary greatly between games; for example, the narrative of a game in which NPCs can be killed must be more flexible than one in which NPCs always live. This variation makes it difficult to compare flexibility between games. For SN-GEN, I will focus on its flexibility to the limited player interaction within Sprite Quest.

In order to evaluate the flexibility of SN-GEN, I have run tests with varying amounts of simulated interference. The case with no player interference corresponds to the Brownie quest in Sprite Quest, in which the player always completes quests as given and therefore follows the author's story. On the other hand, the cases with a lot of player interference correspond to the Imp quest in Sprite Quest, in which the player performs the opposite action than the one which follows by the intended plot. As in the actual Sprite Quest game, it is not possible for the player to reverse every single action; at least a few are carried out correctly. There are two results that I hope will be shown by these tests:

1. The amount of player interference should be correlated with the speed at which the narrative resolves. In particular, the runthroughs with more player interference should take longer to reach the story goal, since the player is working against those goals. If there was little difference between the conditions, it would indicate that the player is not having much effect on the narrative.

2. Despite the player interference, the runthroughs should still reach the author's goal in a reasonable number of steps. If player interference prevents the narrative from reaching its goal, then the narrative is not very resilient.

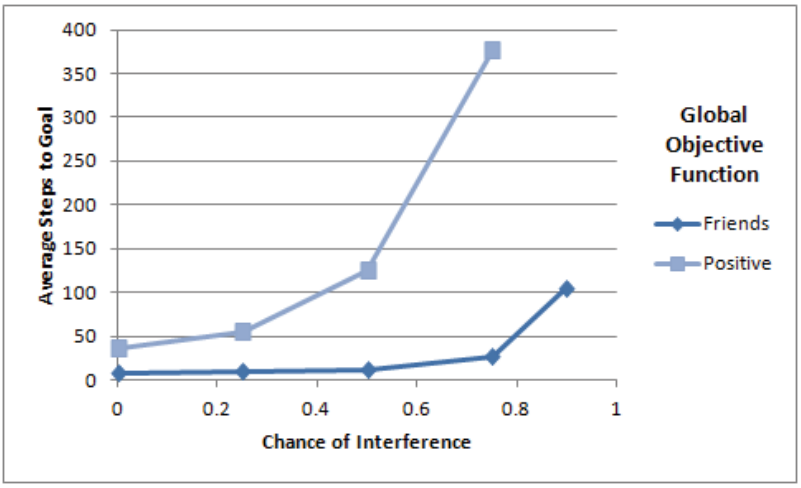

Figure 8: The average number of steps to the story goal over 50 trials, with a certain percentage of actions randomly reversed so that the opposite action is carried out instead. There were 8 agents, and the target steps number was 20 for Friends and 10 for Positive.

Figure 8 shows the time taken for the narrative to resolve with varying amount of player interference. The results clearly support the first hypothesis, since a larger chance of interference corresponds to a higher number of steps. The resilience discussed in the second hypothesis seems to be different depending on the type of objective function. The *Friends* goal resolves in a fairly low amount of steps even when there is about a 50% chance of interference. On the other hand, the performance of the *Positive* goal worsens much more quickly. It makes sense for a goal that depends on all agents to be more susceptible to randomized player interference than a goal that cares about two agents only.

There are additional factors to consider when analyzing flexibility. First of all, this test used interference in which the player always did the opposite of what was expected. If SN-GEN were used in a more open-ended game, player interference might not always directly contradict the intended narrative. In that case the resilience would probably be higher. Another variable that might be different in a real use of SN-GEN is the consistency of interference. The results of a period of high interference followed by low interference would form a pattern not seen in this experiment. Because the weight of the story goal increases over time in SN-GEN, the narrative might resolve quickly during the low interference phase, showing more flexibility.

5.3 Comparison with existing systems

SN-GEN is primarily an emergent narrative system, so it may be useful to compare it to other emergent narratives. It shares many features with "The Prom"; both are emergent narratives based around a social network [McCoy et al., 2010]. The Prom has a far more complex system of social relationship; however, SN-GEN incorporates author control. If SN-GEN were expanded to include more complexity, the result might be comparable to The Prom but with a stronger narrative.

Brenner's Multiagent Planning Language, on the other hand, is very different from SN-

GEN [2010]. The planning algorithms behind MAPL are more complex than the relatively simple social network analysis used in SN-GEN, and the result is a more complex narrative. However, it is not clear how well MAPL works in a live, graphical game; the only examples of it are text-based. The implementation of SN-GEN in Sprite Quest indicate that in a live game, it can generate a detailed story without using complex planning techniques.

SN-GEN is also designed to incorporate authorial influence, so how does it compare to other top-down systems? A limitation of most top-down systems is that they still require a large amount of authoring. For example, planning for different possibilities in Façade required the authors to describe rules to handle a large number of player actions [Mateas and Stern, 2003]. In SN-GEN, on the other hand, the player's actions simply change the game state, which affects agent's choice of actions, and no further handling is necessary.

Another difficulty with top-down systems is how to deal with the player deviating too far from the planned narrative. For example, Mateas et al.'s Automated Story Director is forced to default to an emergent system if the player goes too far off-book [2003]. One of the key features of SN-GEN is that it will continue to follow the desired plot point, no matter what the player does; as the player interferes, the weight of the global story goal gets higher so that agents are always pulled towards the desired state.

SN-GEN does share a limitation with some of the top-down games, including Haunt 2: the player's actions are quite limited [Magerko et al., 2004]. In SN-GEN, this limitation is caused by the state representation; it would be possible to expand GEN to use a different state representation as described in the following section.

5.4 Future work

SN-GEN, as presented here, is quite limited. Social network analysis may lend itself to games with a strong focus on relationships, but as we have seen above, it cannot cover a variety of scenarios that designers may wish to include in games. Therefore, an obvious pathway for exploration is developing other ways of representing the game state which would allow for more actions using GEN. Since the process of choosing a new action can quickly become too expensive for a live game, it is important that the objective functions be as simple. One possible method of dealing with this limitation is to minimize the number of times or the extent to which the objective function is evaluated. For example, an agent could take note of what parts of the network have changed and re-evaluate those parts only. Improvement could come from the implementation of an easier heuristic which eliminates possible actions that definitely will not be chosen.

Another extension to GEN would be to incorporate multiple levels of narrative generation. For example, a high-level story generator could be used to choose a sequence of story goals, which GEN is then used to fulfill. Alternatively, agents could apply more sophisticated techniques, such as planning, in order to complete their goals. This would improve the performance on clique-based goals and enable the use of more complex goals. Combining GEN with other techniques could result in much more sophisticated and interesting stories than SN-GEN alone.

6 Conclusion

The previous work in narrative AI has resulted in a lot of great tools and techniques for interactive narrative generation which vary a lot in complexity and in the balance between story integrity and player freedom. Either an emergent approach or a top-down approach might be more useful depending on the type of game; however, an ideal narrative generation system would combine features from both to create a fully interactive but still cohesive narrative. SN-GEN was an attempt to integrate author control without compromising player freedom. Although its expressive power is limited, SN-GEN strikes a balance between responding to player input and following the intended story. The game Sprite Quest shows that SN-GEN can be used to produce a detailed series of interactions which form the backdrop to the gameplay.

References

Aylett, R. (1999). Narrative in virtual environments - towards emergent narrative. In *AAAI Technical Report*.

Brenner, M. (2010). Creating dynamic story plots with continual multiagent planning. In *Proceedings of IEEE Computer Animation*.

Freidman, N. (1955). Forms of the plot. *Journal of General Education*, (8):241–253.

Krebs, V. (2011). Social network analysis, a brief introduction. http://www.orgnet.com/sna.html.

Lucas, S. M., Mateas, M., Preuss, M., Spronck, P., and Togelius, J. (2012). Artificial and computational intelligence in games (dagstuhl seminar 12191). *Dagstuhl Reports*, 2:43–70.

Magerko, B., Laird, J. E., Assanie, M., Kerfoot, A., and Stokes, D. (2004). Ai characters and directors for interactive computer games. *American Association for Artificial Intelligence*.

Mateas, M. and Stern, A. (2002). A behavior language for story-based believable agents. *Intelligent Systems, IEEE*, 17(4):39–47.

Mateas, M. and Stern, A. (2003). Integrating plot, character and natual language processing in the interactive drama facade. In *1st International Conference on Technologies for Interactive Digital Storytelling and Entertainment*.

McCoy, J., Treanor, M., Samuel, B., Tearse, B., Mateas, M., and Wardrip-Fruin, N. (2010). Authoring game-based interactive narrative using social games and Comme il Faut. In *Proceedings of the 4th International Conference & Festival of the Electronic Literature Organization*.

Peinado, F. and Gervás, P. (2004). Transferring game mastering laws to interactive digital storytelling. *Technologies for Interactive Digital Storytelling and Entertainment*, pages 48–54.

Riedl, M., Thue, D., and Bulitko, V. (2011). *Game AI as Storytelling*. Springer Verlag.

Riedl, M. O., Stern, A., Dini, D. M., and Alderman, J. M. (2008). Experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications*.

Thue, D., Bulitko, V., Spetch, M., and Webb, M. (2010). Socially consistent characters in player-specific stories. In *Proceedings of the Sixth Artificial Intelligence and Interactive Digital Entertainment Conference*.