A Computational Model of Narrative Generation for Suspense

Yun-Gyung Cheong and R. Michael Young

Liquid Narrative Group
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
ycheong@ncsu.edu, young@csc.ncsu.edu

Abstract

Although suspense contributes significantly to the enjoyment of a narrative by its readers, its role in dynamic story generation systems has been largely ignored. This paper presents Suspenser, a computational model of narrative generation that takes as input a given story world and constructs a narrative structure intended to evoke the desirable level of suspense from the reader at a specific moment in the story. Our system is based on the concepts that a) the reader's suspense level is affected by the number of solutions available to the problems faced by a narrative's protagonists, and b) story structure can influence the reader's narrative comprehension process. We use the Longbow planning algorithm to approximate the reader's planning-related reasoning in order to estimate the number of anticipated solutions that the reader builds at a specific point in the story. This paper discusses our preliminary results and concludes with suggestions for further study.

Introduction

Suspense contributes significantly to the enjoyment of a narrative by its readers (Brewer and Lichtenstein, 1982; Alwitt, 2002). Suspense is the feeling of excitement or anxiety that audience members feel when they are waiting for something to happen and are uncertain about a significant outcome.

This paper presents a computational model of suspense, exploring the concept that a reader's suspense level is affected by the number of solutions available to the problems faced by a narrative's protagonists (Gerrig and Bernardo, 1994; Comisky and Bryant, 1982). In particular, our model focuses on plot-suspense, which differs from action-suspense in that the former is generated from plot development and the latter is evoked by the reader observing physical action scenes (such as car chases in film). For viewers to feel plot-suspense from a narrative, the story should unfold in a specific manner. Brewer and Lichtenstein (1982) suggest that a suspenseful story should be structured as a sequence involving a) an initiating event that is leading to a significant outcome, b) intervening events, and c) the outcome.

Our approach attempts to manipulate the level of suspense experienced by a story's reader by elaborating on the story structure — making decisions regarding what story elements to tell and when to tell them — that can influence the reader's narrative comprehension process. To this end, we make use of a computational model of that comprehension process based on evidence from previous psychological studies (Brewer, 1996; Gerrig and Bernardo, 1994; Comisky and Bryant, 1982). To generate suspenseful stories, we set out a basic approach built on a tripartite model, adapted from narrative theory, that involves the following elements: the *fabula*, the *sjuzhet*, and the discourse (Rimmon-Kenan; 2002). A *fabula* is a story world that includes all the events, characters, and situations in a story. In our approach, the *fabula* is represented as a plan structure generated by Crossbow—a hierarchical, partial-order causal link planner based on the Longbow planning system (Young et al., 1994). A *sjuzhet* is a series of events selected from the *fabula* and an ordering over those events indicating the order in which they are to be presented to readers. The final layer, a discourse, can be thought of as the medium of presentation itself (e.g., text, film). Although not directly discussed in this paper, discourse is important for the effective presentation of a story for the reader (Callaway and Lester, 2002).

We present Suspenser, a framework that constructs a narrative structure (i.e., *sjuzhet*) from a given story world (i.e., *fabula*) intended to evoke the given level of suspense (i.e., either high or low) from the reader. We assume in the work described here that the stories we deal with all contain conflict. For example, characters' individual goals may be negations of each others', or the plans formed by characters to achieve their goals may interfere with the plans of other characters. While other dramatic devices such as the prolonging of resolution are also useful in crating suspense, we focus here on suspense that arises as a result of users' consideration of these conflicts and their consequence on the protagonist's goals. We expect that Suspenser can be incorporated into various narrative-centered applications such as games and interactive dramas in virtual worlds where the effect of suspense serves the user's qualitative experience.

Related Work

Automated story generation has been extensively studied, with applications ranging from computer games to education and training (Cavazza, 2002; Riedl and Young, 2004; Mateas and Stern, 2003). However, little attention has been paid to computational story generation dealing with suspense. As a result, this section reviews suspense related psychological research and interactive narrative systems that focus on the user's experience of the story.

Suspense is characterized by Vorderer (1996) as a phenomenon associated with three dimensions—type of text, the reader, and the reader's emotional process. The first dimension, type of text, suggests that the reader will experience a higher level of suspense when reading text describing physical actions rather than when reading text describing a character's thought and emotion. Vorderer hypothesizes that the second factor, the readers, affects the level of suspense that a text evokes based on their inter-individual factors (e.g., age, gender) and intra-individual factors (e.g., physical locations, moods). The last dimension, the reader's emotional process, influences the experience of suspense: a reader will imagine a preferred outcome for a character within a story that she identifies with, and consequently, if the preferred outcome looks unachievable, the reader feels suspense.

Previous psychological studies have explored story structure regarding suspense (Brewer, 1996; Brewer and Lichtenstein, 1982). Brewer (1996) suggests the relevance of structural-affect theory, claiming that affective states in the reader are provoked by arranging the temporal ordering of the events underlying a story world. The theory explains that suspense could be evoked by presenting the events of a story chronologically to the reader while surprise and curiosity could be caused by hiding a critical fact or event early in the story world and disclosing it later in the text. Film directors also arrange the filming of story elements to manipulate the reader's beliefs for evoking suspense (Gerrig, 1996).

A number of AI researchers have presented user-centric views to story generation that are closely related to the approach we take here. Bailey (1999) suggests an approach to generating stories considered interesting by the reader by accessing the reader's knowledge-base. His system follows a cycle composed of four stages. The first phase manipulates (i.e., generalize, specialize, detach, join) the current reader's knowledge-base to generate candidates for the next story segment. The second measures the effect of each candidate story unit based on the reader's expectations and questions with the story-so-far. The third step chooses a segment resulting in a good pattern of question and expectation based on a storiness heuristics. Last, the selected story unit serves as the next story segment, and the reader's model is updated accordingly. Although his research exploits the reader's role in story generation, Bailey does not suggest a solution to the formalization of his storiness heuristics; thus, the plausibility of his approach is difficult to gauge.

In an interactive narrative system where the user participates as a character in the story, choices made by the user influence the story development. To provide the user with an effective experience while she interacts with a virtual story world that the Oz system (Kelso et al., 1993) presents, Weyhrauch (1997) developed the Moe architecture, which views the interaction between the user and the system as a sort of an adversarial search. Moe consists of two main components: an evaluation function and an adversarial search. The evaluation function rates the quality of a sequence of user moves generated in the course of her experiencing the story world. The result of Weyhrauch's experiment shows that the evaluation function correctly approximates two human experts' aesthetic evaluation on a user's experience. In addition, the article reports the effectiveness of the shallow searching method from the result of the experiments with simulated users parameterized by confirmation to the system's guidance. And yet, the universal application of the shallow searching method is questionable. Nelson and Mateas (2005) report that the Moe system has little impact on improving the user's experience in an interactive world based on an interactive mystery fiction *Anchorhead*. In the article, they ascribed the dissatisfactory result to the shallow searching method that the Moe architecture employs.

While Moe directs the user towards an ideal sequence of moves, DEFACTO (Sgouros, 1999) determines the subsequent event and its desirable outcome responding to the user's current choice. DEFACTO uses a rule-based plot manager to create story lines that conform to the concept of plot of initial situation, a climax, a conflict and a resolution, as first described by Aristotle. The plot manager builds a plot to provide dramatic experience for the user, who plays the role of a protagonist in a story. The plot manager first constructs possible sequences of actions for each character's goal and roles; its second stage selects action sequences that follow one the four dramatic patterns (i.e., lifeline, rising-complication, reversal-of-fortune, and irony); then it selects one dramatic situation that gives the user a greater degree of participation as the next plot element. These phases of generation and evaluation repeat sequentially until no new interesting interactions are found. Finally, the resolution phase determines the outcome of each action, success or failure. Sgouros' research proposes a user-centric view to story generation. Particularly, his evaluation phase conforms to the observations by Kelso et al (1993), who report that a user is strongly engaged when she actively participates in the story even when the audience observing the action might feel bored. However, this plot manager is designed for an average type of user who acts as a protagonist in the story. Consequently, it is not clear how the system handles a multi-user environment or an environment where the user serves as a viewer.

As a theoretical model of affective storytelling, Platts, Blandford, and Huyck (2002) describe a reasoning-based approach that produces a twisted story. When a seed story is given as input, their approach first divides the story into

episodes and identifies its climax. By the multiple application of backward reasoning to the climax, the system generates two distinct versions of the seed story, an overt story and a concealed story. Then, a twist phase is created by reasoning forward from the climax episode of the concealed story that explains the transition from the overt story to concealed story. Finally, the overt story and twist phase episodes are assembled into a complete final story. Posing a twisted story generating process as synthesizing two different stories that share a climax, their study identifies essential processes for twisted story generation. However, those processes are only partially specified, and their implementation is under development. As a result, it is difficult to measure the effectiveness of their approach.

As discussed above, while psychological research emphasizes the reader's cognitive processing in his experience of suspense, the role of a reader in most work on computational story generation has been restricted. Therefore, current story generation systems have not significantly addressed the effect of suspense. To bridge this gap, we present Suspenser, a system that creates a suspenseful story structure by modeling the reader's planning-related reasoning process using a planning technique.

The Suspenser Architecture

This section describes the Suspenser architecture as illustrated in Figure 1. Our system takes three elements as input: a *fabula*, a desired suspense level (i.e., either high-suspense or low-suspense), and a given point *t* in the story plan that corresponds to the point where the reader's suspense is measured. Then Suspenser determines the *sjuzhet*, both the content and, to a given extent, the ordering of the discourse to be used to convey the story up to *t* to a reader. In our system, *fabula* and *sjuzhet* are defined in terms of planning as follows.

Definition 1 (*Fabula*) A *fabula F* is a tuple <*S, B, O, C, D*> where *S* is a series of plan steps, *B* is a set of binding constraints, O is temporal ordering information, *C* is a list of causal links, and *D* is a list of decompositional links. *S* is represented as $<s_1, s_2, \ldots, s_n>$ where s_i is an instantiation of a plan operator contained in a plan library. A plan operator *op* is a tuple <*N, P, E*> where *N* is a unique string, *P* is a set of preconditions representing just those conditions that must hold for *op* to be able to happen, and *E* is a set of effects denoting just those conditions that are changed by the action's successful execution. A causal link is represented as $(s_i \rightarrow s_j; e)$, notating a plan step s_i establishes *e*, a precondition of a subsequent step s_j. A decompositional link is shown as $(s; s_1, s_2, \ldots, s_n)$, interpreted as an abstract plan step *s* is decomposed into sub-actions $s_1, s_2, \ldots, s_n$. Temporal ordering information is denoted as $(s_i < s_j)$ where s_i precedes s_j. A binding constraint is described as $<s_i; (p, c)>$ where a plan step s_i binds constant *c* for the step's parameter *p*.

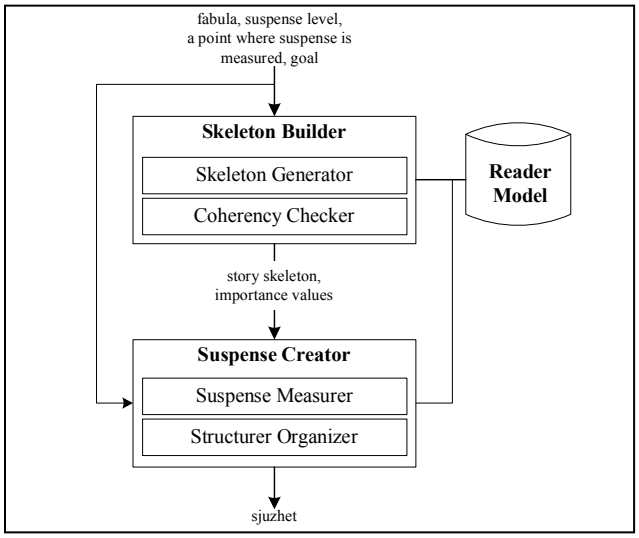

Figure 1. The Suspenser Architecture

Definition 2 (*Sjuzhet*) A *sjuzhet Z* is a tuple <*F, S, T*> where *F* is a *fabula*, *S* is a subset of the plan steps of *F*, *T* is presentation ordering of the plan steps in *S* to be presented to the user. Presentation ordering information is denoted as $(s_i < s_j)$ where s_i precedes s_j. *Z* uses the ordering information of *F*, however, when ordering information of *T* conflicts with ordering information of *F*, the ordering of *T* takes precedence over the ordering of *F*.

When a *fabula* and a *sjuzhet* are represented as planning structures, the task of Suspenser is to take a *fabula* as input and construct its *sjuzhet* which enables the reader to infer a minimum number of complete plans for the protagonists' goal, following the psychological research on suspense (Gerrig and Bernado, 1994; Comisky and Bryant, 1982). In addition, we require that the resulting *sjuzhet* shall be read as a coherent story that represents the input *fabula.*

To produce a *sjuzhet* meeting these requirements, we develop a framework composed of two phases: a skeleton building step and an additional story element identification step. In the skeleton building step, Suspenser identifies the *skeleton* of the *fabula*—a partial plan that specifies its plan steps as a set of core story events that cannot be eliminated without harming the understandability of a story—by rating each individual event's importance based on the event's causal relationship to the protagonists' goals. In the second phase, Suspenser finds additional plan steps α which confine available solutions for the protagonist's goal and plan steps β which help the reader find more solutions for the goal within her cognitive limit. Finally, Suspenser composes the content of the *sjuzhet* by adding α and by adding a temporal order of delivering β after *t*.

The following subsections describe each of the three components of Suspenser as in Figure 1, with special attention to the suspense creator which is crucial for the affect of suspense. We discuss the reader model first because the model is used by both the skeleton builder and the suspense creator.

The Reader Model

The reader model represents the individual's reasoning algorithm, reasoning capacity, knowledge, and preference. For the reasoning algorithm in this paper, Crossbow (Young et al., 1994) is used. As a form of reasoning limit, an integer is used to constrain the number of nodes to be searched in planning. To represent the reader's knowledge, a set of operators is defined as a plan library. Each operator has its unique name, a set of preconditions and effects, and a set of variables that shall be instantiated in the planning process. Preference in the current system stores the user's heuristic function for planning process, and the reader's needs such as parameters that control the content selection processes or a preferable story length.

In order to model the reader's inference process and anticipation of the protagonists' success, Suspenser uses Crossbow to model the reader's plan-related reasoning processes. Prior work has provided strong evidence that human task reasoning is closely related to partial-order planning algorithms (Rattermann, 2001) and that *refinement search* (Kambhampati et al., 1995), the type of plan construction process performed by Crossbow, can be used as an effective model of the plan reasoning process (Young, 1999).

Refinement search (Kambhampati, 1995) views the planning process as search through the plan space represented as a directed acyclic graph composed of nodes denoting partial plans. In our approach, the root node of the graph is a partial plan given from the skeleton builder or the suspense creator. The leaf nodes of the graph are either complete plans without flaws or plans with flaws that cannot be repairable due to inconsistency in the plan; internal nodes are partial plans with a number of flaws.

A flaw in Crossbow is either a precondition of some step that has not been established by a prior step in the plan, or a causal link that is threatened (i.e., undone) by the effect of some other step in the plan. In the graph, a child node is a refinement of its parent node to repair a single flaw in the parent plan. When the flaw is an open precondition, a causal link is established from either an existing step in the plan or an instantiated operator in the plan library which has an effect that can be unified with the precondition; in the second case, the instantiated step is added to the parent plan. When the flaw is a threatened causal link, a temporal ordering (i.e., either demotion or promotion) to resolve the threat is added or binding constraints are added to separate the threat involved steps so that no conflicts arise. If the flaw is an abstract step, then the step is decomposed into a series of primitive plan steps as encoded in a decomposition schema.

The Skeleton Builder

This section describes the skeleton builder, which determines important events based on the user's knowledge, and produces a partial plan that specifies those events as its plan steps. The skeleton builder consists of two components: the skeleton generator and the coherency checker. The skeleton generator extracts a series of important events of the story, i.e., a skeleton, and then the coherency checker tests the skeleton to ensure that its content can be understood as an integral story.

To generate a candidate skeleton, the skeleton generator rates the importance of each event based on a method for extracting important actions that are likely to be included in the story recall, devised by Trabasso et al. (1984). To determine an individual story event's importance, their approach counts the number of causal relationships with other steps in the narrative and measures each event's importance by analyzing its role in a series of actions in a story that are causally related. Adapting their approach, the skeleton builder approximates causal relationships by counting the number of incoming and outgoing causal links of a plan step and measuring the qualitative importance of events which are determined by their roles in the plan. We define three important roles of events in a story plan: an opening act, a closing act, and a motivated act. An opening act is the first action in the plan. A closing act is the last action that occurs in the story. Motivated acts are actions that establish a literal of the goal state. We apply a simple linearization routine to the *fabula* to detect the opening act and the closing act in a plan. After computing each event's importance, the top *N* events are selected. The value for *N* can be set as either an integer or a ratio against the total number of actions in the plan. From these chosen events the system builds a skeleton, a partial plan that specifies those events as its plan steps.

Once an initial candidate skeleton is generated, the coherency checker tests whether the skeleton is coherent from the reader's perspective using an algorithm which is a cycle composed of two phases: coherency check and event selection. The coherency check step uses the reasoning algorithm in the reader model—Crossbow—to find complete plans to achieve the protagonist's goals which are consistent with the skeleton candidate. If such a plan is found, the story skeleton is coherent and the program exits. Otherwise, an event in the *fabula* which was not selected as a skeleton with the highest importance value is selected and added to the candidate. Then, the coherency check phase begins again. Finally, the story skeleton and the importance rate for each event of the input *fabula* are passed to the suspense creator.

This skeleton builder in principle follows the Cooperative Plan Identification (CPI) architecture, a computational model that generates concise textual descriptions of plans developed by Young (1999). Like the CPI model, the skeleton builder extracts a partial plan that enables the recipient to generate a complete plan. The skeleton builder, however, differs from CPI in two ways. First, the skeleton builder considers the qualitative importance of an event, which is not considered in CPI. Second, CPI requires the hearer's complete plan similar to the original plan, which is not demanded by the skeleton builder. Those distinctions are due to their different domains: narrative generation for enjoyment and instruction generation for performing specific tasks.

Initialization: Z = <F, S, O> where F is the input fabula, S = K where K is a set composed of event steps in the skeleton, O = {}

Termination: If *S* is empty or no candidates satisfying the following conditions are found, then return *Z*.

Event Selection:

1) Select an action *e* contained in F but not included in *S* which has the greatest positive *potential suspense*. If several candidates are found, non-deterministically select an action with the greatest *importance value*.
2) If the suspense level from a partial plan which has all the plans steps *(S + e)* is greater than the suspense level with a partial plan which has all the plan steps in *S*, then add *e* to *S*.

Presentation Order Arrangement:

1) Select an action *e* in *K* with the smallest negative *potential suspense*. If several candidates are found, non-deterministically select an action with the highest *importance value*.
2) If the suspense level from a partial plan which has all the plans steps in *(S – e)* is greater than the suspense level with a partial plan which has plan steps in *S, then add a temporal order (t < e)* to *O*.

Figure 2. Algorithm for content selection and presentation ordering in the high-suspense mode

The Suspense Creator

The suspense creator takes as input the story skeleton and importance value for each action of the input *fabula* from the skeleton builder. The function of the suspense creator is to construct the *sjuzhet* (content and presentation order) evoking the intended suspense level from the reader at *t*, the target point when the reader's suspense level is measured. The suspense creator consists of two components: the structure organizer and the suspense measurer. When the *sjuzhet* is initialized with the given story skeleton, the structure organizer updates the *sjuzhet* with the story elements that can influence the reader's suspense level in cooperation with the suspense measurer which returns the corresponding suspense level for a given content.

The overall algorithm that the suspense creator performs to produce a highly suspenseful story is illustrated in Figure 2. In the algorithm, we introduce the term *potential suspense* that refers to the amount of each event's contribution to the suspense level increase, which is computed using Heuristic function 2, as described in the next section. The algorithm consists of two steps: an event selection and an ordering arrangement. In the first phase, the event selection, an action with the greatest potential suspense is chosen as α, and creates a partial plan *P* composed of α along with the plan steps of the skeleton. If the suspense level from *P* is greater than the suspense from the skeleton, then the current skeleton is replaced with *P*. This process continues until there is no candidate for α. When the first phase terminates, the system specifies the output *sjuzhet* as the current skeleton. In the ordering arrangement phase, an action in the initial skeleton sent from the skeleton builder which has the lowest potential suspense is chosen as β. Then, the suspense creator builds a partial plan *P* composed of the plan steps of the initial skeleton excluding β. If the suspense level from *P* is lower than the suspense from the skeleton, the system modifies the *sjuzhet* to reflect the presentation ordering of $(t < \beta)$, which means that telling of β is deferred after *t*. This process repeats for a predefined number, which shall be set as a small enough so that the coherency of the skeleton can be maintained.

The algorithm in the low-suspense mode is similar to that in the high-suspense model. However, as opposed to the high-suspense mode, the first phase selects an action with the lowest potential suspense as α, and checks if the suspense level is lowered by adding α to the skeleton. The second phase finds an action with the highest potential suspense in the skeleton as β, and checks if this ordering lowers the suspense level.

To explain our algorithm in the context of a story, we take an example from a film. In the ending of *Back to the Future*, Marty McFly (acted by Michael J. Fox) came back to 1985, saw the killers of Dr. Brown driving in the direction toward him, and re-witnessed their shooting at him. A moment later, however, the audience realized that Dr. Brown was still alive because of the bullet-proof vest that he was wearing. In this illustration, two noticeable film devices exemplify our algorithm. First, scenes that seem to jeopardize the protagonist's goal (e.g., the killer's driving toward the doctor) are intentionally shown to arouse suspense from the audience, which can be modeled by the event selection phase of our algorithm. Second, telling the advantageous facts or events to the protagonists is postponed until the outcome is revealed, which corresponds to the task of our ordering arrangement step performs. In the film, if the audience members knew about the bullet-proof vest, they would feel little or no suspense from the illustrated scenes.

In the following subsections, we present two heuristic functions used in the algorithm: a function that measures suspense level from a given partial plan and a function that returns potential suspense for a given step in a plan.

Measuring Suspense Level

In measuring the suspense level on the reader's part, the system follows the notion articulated by Gerrig and Bernardo (1994), in which they view an audience as problem-solvers: an audience will feel an increased measure of suspense as the number of options for the protagonist's successful outcome(s) decreases.

Adopting these models, we devise Heuristic Function 1 for measuring the *level of suspense*; the function computes the reader's suspense level as the inverse of the number of planned solutions for the protagonists' goal using her reasoning algorithm and her plan library within her

reasoning limit. The function sets the minimum level of suspense when no usable solutions are found in her plan space, as evidence in psychological research.

Heuristic Function 1 (*Level of suspense*) The Suspense level function *SL(G, Z, L, P, R)* returns (1/*success(G, Z, L, P, R))* when *success(G, Z, L, P, R)* returns non-zero value where *G* is a set of literals representing the goal of a narrative's protagonist, *Z* is a *sjuzhet*, *L* is a plan library, *P* is a planning algorithm, *R* is an integer representing a reasoning bound, and *success(G, Z, L, P, R)* returns the number of paths to make *G* true with given *Z* and *R*. When *success(G, Z, L, P, R)* returns 0, *SL(G, Z, L, P, R)* returns 0.

Measuring Potential Suspense for an Action

In computing the potential suspense of an action's effect, we consider the action's all possible causal relationship to accomplishing the protagonist's goal from the reader's point of view. We devise Heuristic Function 2 to compute the potential suspense for an action by counting the number of its effects that negate the protagonist's goal and the number of its effects that unify the goal under the assumption of the audience's partial knowledge. As an illustration, Figure 3 shows that the action *A* has an effect *~g*, which is the negation of the goal literal *g*. We call this type of a temporary threat as a *threatening link*, referring to an action's effect negating another step's precondition in the plan. In contrast, the suspense creator establishes a *supporting link* when an effect of an action unifies with a precondition of an action in the plan. One effect can have multiple threatening links or supporting links in a single plan. The potential suspense of an effect is computed as the supporting link summation subtracted from the threatening link summation as formalized in Heuristic Function 3.

Heuristic Function 2 (Potential Suspense of an action) *h(a, p)* returns the summation of *ps(e, a, p)* where *ps(e, a, p)* is the potential suspense of the effect *e* of the action *a* in the plan *p*.

$$h(a, p) = \sum_{e \in effects(a)} ps(e, a, p)$$

Heuristic Function 3 (Potential Suspense of an effect) *ps(e, a, p)* returns potential suspense of an effect *e* of an action *a* in a plan *p*, which is the summation of the potential threat of all *e*'s supporting links subtracted from the summation of the potential threat of all *e*'s threatening links as formalized as the following equation.

$$ps(e, a, p) = \sum_{t \in Tlink(e)} \frac{w_t}{dist(d_t, p)} - \sum_{s \in Slink(e)} \frac{w_s}{dist(d_s, p)}$$

Where *Tlink(e)* returns all the threatening links of an effect *e*, *Slink(e)* returns all the supporting links of *e*, w_t and w_s are coefficients, d_l denotes the destination step of the link *l*, and *dist(s, p)* returns ((the minimum number of causal link chains that connect the plan step *s* and the goal state in the plan *p*) + 1).

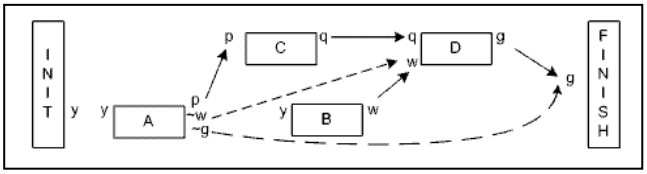

Figure 3. Threatening links in a story plan. A box represents an action, with its preconditions on the left and effects on the right. Solid arrows denote causal links. Dotted arrows are threatening links which represent an action's effect negates a precondition of other actions.

Evaluation

This section examines a pilot study that we carried out to evaluate the effectiveness of stories in terms of suspense generated by the current implementation of Suspenser compared with human created stories.

An Input *Fabula*

To obtain an input to Suspenser, we ran Crossbow to generate a *fabula* plan which involved five characters: the President, an anti-hero Dr. Evil, who plans to assassinate the President, a renowned environmentalist Mr. Greenpeace, and a poor father Tom, who is the father of a six-year old boy named Ben. Crossbow took as input the planning problem, which specifies the initial and goal of the story, and a plan library composed of 17 plan operators, and then returns a complete plan containing: actions for Dr. Evil to assassinate the President, and actions for Mr. Greenpeace to save the earth, and actions for Tom to get Ben a Christmas gift, and actions to keep the President alive. The resulting plan consisted of partially ordered 25 steps which were manually linearized, and the plan was realized as text as in Figure 4a using a simple template matching technique which mapped one plan step into a single sentence.

Four *Sjuzhets*

For our pilot study, we prepared four *sjuzhets*: two stories by Suspenser and two stories by humans. Since the current implementation of Suspenser lacks in the reader model, the pilot study was to test if the heuristic functions 2 and 3, predicting the potential suspense of an action and an effect, were effective in identifying story events that manipulate suspense level, with the cooperation of the skeleton. The values of the scaling factors for Heuristic Function 3, estimating the potential suspense of an effect, that were used in this study are as follows: for threatening link $w_t = 2$ and for the supporting link $w_s = 1$. We assigned a greater value for the threatening link coefficient to compensate the supporting strength by the causal links of the plan. To identify a series of events that increases the suspense level, we selected actions with potential suspense greater than a threshold (i.e., -0.3 for this study). In a similar fashion, a set of actions that reduces the suspense level was chosen as actions with potential suspense lower than a threshold (i.e., for this study).

[1] Mr. Greenpeace traveled from the Amazon to the US Capitol. [2] Mr. Greenpeace made a speech about the importance of taking action immediately to save the world. [3] The President announced that he would raise funds to support Mr. Greenpeace's environmental foundation and whoever donated more than million dollars would be invited to the White House for a fund-raising celebration party. [4] Dr. Evil watched the TV and found out that a donation would get him invited to the White House. [5] Dr. Evil donated a million dollars to the White House. [6] The President traveled to the White House. [7] The President invited Dr. Evil to the fund-raising celebrating event. [8] The President gave the promised government financial support to Mr. Greenpeace's foundation. [9] Tom traveled to Dr. Evil's castle. [10] Tom traded his ring for Dr. Evil's toy; as a result, Tom obtained the toy that Ben wanted and Dr. Evil obtained the ring. [11] Tom traveled back to his house, and went up to the Christmas tree. [12] Tom put the toy under the Christmas tree. [13] Ben walked from his room to the Christmas tree. [14] Ben found his Christmas present—the toy that Tom left. [15] Dr. Evil went to a bank to withdraw money from his bank account. [16] Dr. Evil withdrew enough cash from his account to buy a gun and to register a hypnosis class. [17] Dr. Evil traveled to a gun store. [18] Dr. Evil bought a gun. [19] Dr. Evil registered for a hypnosis class to learn how to hypnotize people by waving a shiny object before their eyes. [20] Dr. Evil took a hypnosis class; as a result, he knew how to hypnotize people by waving a shiny object before their eyes. [21] Dr. Evil traveled to the White House. [22] Dr. Evil used the ring of power to put all the Secret Service agents to sleep; as a result, there was no one guarding the president. [23] Mr. Greenpeace traveled to the White House. [24] Dr. Evil fired his gun at the President. [25] At the last moment, Mr. Greenpeace rescued the President by pushing him out of the way.

Figure 4a. The input *fabula*

Mr. Greenpeace made a speech about the importance of taking action immediately to save the world. The President announced that he would raise funds to support Mr. Greenpeace's environmental foundation and whoever donated more than million dollars would be invited to the White House for a fund-raising celebration party. Dr. Evil watched the TV and found out that a donation would get him invited to the White House. Dr. Evil donated a million dollars to the White House. The President invited Dr. Evil to the fund-raising celebrating event. Tom traded his ring for Dr. Evil's toy; as a result, Tom obtained the toy that Ben wanted and Dr. Evil obtained the ring. Ben found his Christmas present--the toy that Tom left. Dr. Evil went to a bank to withdraw money from his bank account. Dr. Evil bought a gun. Dr. Evil registered for a hypnosis class to learn how to hypnotize people by waving a shiny object before their eyes. Dr. Evil traveled to the White House. Dr. Evil used the ring of power to put all the Secret Service agents to sleep; as a result, there was no one guarding the president. Dr. Evil fired his gun at the President.

Figure 4b. A suspenseful story generated by Suspenser

Story generator	**Suspense Level**				Total
	No	A little	Moderate	A lot	
Human	4	4	4	1	13
H-Suspenser	2	7	5	0	14
L-Suspenser	5	4	3	0	12
Total	11	15	12	1	39

Table 1. Collected data for each story category. H-Suspenser stands for Suspenser in high-suspense mode and L-Suspenser stands for Suspenser in low-suspense mode

The thresholds used in this study are adjusted from a number of informal experiments. From this setting, the current system produced two stories, one shown in Figure 4b in high-suspense mode and a set of <#2, #8, #10, #12, #14, #15, #18, #19, #22, #24> in low-suspense mode.

To obtain human generated stories, we recruited one master student majoring in English and one PhD student in computer science at North Carolina State University. They were presented the text in Figure 4a and were asked to select a series of sentences for suspense except the last sentence (#25), which reveals the story outcome. We did not constrain the number of sentences that they selected.

Procedure

We performed a pilot study with 39 undergraduate students ranging in age from 18 to 29 years old, all recruited from the North Carolina State University. They majored in various fields, including biology, mathematics, social work, political sciences, etc. Each subject individually participated in the study by accessing to a web site that contains a paragraph describing the background and the goal of each character in the story. They were then asked to read text of one of the four *sjuzhets* which is randomly selected. After reading the text, they were asked to answer their suspense levels from the story on a four-scale basis.

Results and Discussion

Table 1 shows the number of responses for each story category. For convenience, H-Suspenser stands for Suspenser in high-suspense mode and L-Suspenser stands for Suspenser in low-suspense mode. As the responses for each category in Table 1 indicates, H-Suspense slightly outperforms a human in three suspense levels (i.e., no suspense, a little suspense, moderate suspense) and outperforms L-Suspenser in all four suspense levels within a relatively large margin.

Further, we used the chi-square test to discover the relationships of suspense levels between the stories. Although the result is not statistically significant due to the small sampling size, the chi-square values did indicate that the two data sets of H-Suspenser vs. human have more similarity than the sets of H-Suspenser vs. L-Suspenser. The data supports the claim that our heuristic functions and the skeleton builder are effective in identifying events of a story that manipulate the affect of suspense from human readers.

Conclusion

This paper describes our computational model for constructing a story structure (i.e., content and presentation order) of a given story plan which manipulates the suspense level that the reader experiences at a specific point in the story. Our model first extracts a coherent summary of the input story to be used as the content of a story structure, and completes the structure by adding story elements that control the suspense level using heuristic functions. For both components, the reader's planning-related reasoning process is modeled using a hierarchical causal link planning algorithm.

The result from a pilot study, testing the functionality of the current implementation of Suspenser, suggests that our model is effective in selecting story elements that contribute to the reader's suspense level. We expect that the full-scale system will yield a consistent result.

Our future work will focus on the extension of our model to interactive environments by expanding the replanning techniques we already looked at (Riedl, Saretto, and Young, 2003; Harris and Young, 2005). Furthermore, we hope that this work will motivate research on affective story generation for providing various emotions for users.

References

Alwitt, L. F. 2002. Suspense and Advertising Response. *Journal of Consumer Psychology*, 12 (1), 35-49.

Bailey, P. 1999. Searching for storiness: Story-generation from a reader's perspective. In M. Mateas & P. Sengers (Eds.) *Narrative Intelligence: Papers from the 1999 AAAI Fall Symposium*, 157-163, Menlo Park, CA.

Brewer, W. F. 1996. The nature of narrative suspense and the problem of rereading. In P. Vorderer, H. J. Wulff, & M. Friedrichsen (Eds.), *Suspense: Conceptualizations, theoretical analyses, and empirical explorations*, 107-127, Mahwah, NJ: Erlbaum.

Brewer, W. F. and Lichtenstein, E.H. 1982. Stories Are to Entertain: A Structural-Affect Theory of Stories, *Journal of Pragmatics*, 6 (5-6), 437-86.

Callaway, C. B. and Lester, J. C. 2002. Narrative Prose Generation. *Artificial Intelligence*, 139, 213–252.

Cavazza, M., Charles, F., and Mead, S. J. 2002. Character-Based Interactive Storytelling. *IEEE Intelligent Systems, 17*(4), 17-24.

Comisky, P., and Bryant, J. 1982. Factors involved in generating suspense. *Human Communication Research* 9(1), 49-58.

Gerrig, R. J. 1996. The Resiliency of Suspense. In P. Vorderer, H. J. Wulff, & M. Friedrichsen (Eds.), *Suspense: Conceptualizations, theoretical analyses, and empirical explorations*, 93-105, Lawrence Erlbaum Associates.

Gerrig, R., and Bernardo, D. 1994. Readers as problem-solvers in the experience of suspense. *Poetics*, 22, 459-472.

Harris, J. and Young, R. M. 2002. Proactive Mediation in Plan-Based Narrative Environments. In *Proc. of IVA 2005*.

Kambhampati, S., Knoblock, C. A., and Yang, Q. 1995. Planning as Refinement Search: A Unified Framework for Evaluating Design Tradeoffs in Partial-Order Planning. *Artificial Intelligence*, 76(1-2), 167-238.

Kelso, M. T., Weyhrauch, P., and Bates, J. 1993. Dramatic presence. Presence: *The Journal of Teleoperators and Virtual Environments*, 2(1), 1-15.

Nelson, M. and Mateas M. 2005. Search-Based Drama Management in the Interactive Fiction Anchorhead. In *Proceedings of AIIDE 2005*.

Platts, J., Blandford, A., & Huyck, C. 2002. Awaiting the Sensation of a Short, Sharp Shock: Twist-Centred Story Generation By Transformation. In *Proc. of the AISB'02 Symposium on AI & Creativity in Arts and Science*, 89-97.

Rattermann, M. J., Spector, L., Grafman, J., Levin, H. and Harward, H. 2002 Partial and total-order planning: evidence from normal and prefrontally damaged populations, *Cognitive Science*, 25(6); 941-975.

Riedl, M., Saretto, C. J. and Young, R. M. 2003. Managing interaction between users and agents in a multiagent storytelling environment. In *Proc. of AAMAS '03*, 741-748.

Riedl, M. O. and Young, R. M. 2004. An intent-driven planner for multi-agent story generation. In Proceedings of *AAMAS 2004*, 186-193.

Rimmon-Kenan, S. 2002. Narrative Fiction: Contemporary Poetics. New York: Metheun, Routledge.

Sgouros, N. M. 1999. Dynamic Generation, Management and Resolution of Interactive Plots. *Artificial Intelligence*, 107(1), 29-62.

Trabasso, T. and Sperry, L. L. 1985. Causal Relatedness and Importance of Story Events. *Journal of Memory and Language*, 24, 595-611.

Vorderer, P. 1996. Toward a psychological theory of suspense. In P. Vorderer, H. J. Wulff, & M. Friedrichsen (Eds.), *Suspense: Conceptualizations, theoretical analyses, and empirical explorations*, 233-254, NJ: Erlbaum.

Weyhrauch, P. 1997. *Guiding Interactive Ficiton*. Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Young, R.M., Pollack, M.E., and Moore, J.D. 1994. Decomposition and causality in partial-order planning. In *Proceedings of AIPS 1994*, 188-193.

Young, R. M. 1999. Using Grice's Maxim of Quantity to Select the Content of Plan Descriptions, *Artificial Intelligence*, 115, 215-256.