Story Planning with Vignettes: Toward Overcoming the Content Production Bottleneck

Mark O. Riedl and Neha Sugandh

School of Interactive Computing, Georgia Institute of Techology
85 Fifth Street NW, Atlanta, Georgia 30308, USA
{riedl; nsugandh}@cc.gatech.edu

Abstract. Storytelling is prominent part of the daily lives of humans. Entertainers, educators, and trainers often concern themselves with the production of novel stories for entertainment, education, and training. However, it is possible for the consumption of story content by end-users to outpace the rate of production of story content. One solution is to instill greater creativity in computer systems in the form of story generation. We present an incremental advancement to planning-based story generation that increases the space of narratives that can be automatically searched in an attempt to make planning-based story generation more creative. The VB-POCL story planning algorithm implements a form of case-based planning that can incorporate vignettes – plot fragments that are a priori known to be "good" – into a narrative planning process. We show that VB-POCL can generate narratives with favorable structural properties that cannot be generated reliably with previous attempts at planning-based narrative generation.

1 Introduction

Storytelling is a pervasive part of our daily lives and culture. Storytelling is particularly prominent in entertainment, where stories can be viewed as artifacts to be consumed by an audience. Story also plays a role in education and training, where stories and scenarios can be used to illustrate and guide. The production of these artifacts – stories and scenarios – is a primary activity in the entertainment industry and also a significant bottleneck in the educational and training industries. In an "on-demand" society, waiting for periodic updates to serial narratives – weekly television series, movie series, and novels – is not considered ideal. Likewise, players of computer games that rely on stories and quests can complete quests faster than designers can create new quests (for a case study, see [1]). How do we handle the situation in which content consumption outpaces content production? One way to overcome the bottleneck of content production is to instill in a computer system the creative ability to generate new content.

Because of the prevalence of story in non-interactive media such as books and movies, as well as interactive media such as computer games, we concern ourselves with the automated generation of narrative content. The issue is whether an automated story generation system can be considered creative enough or skilled enough to be trusted to produce content – stories – that will be experienced by users. More generally, the output of a creative system, such as an automated story

generation system, must be novel, surprising, and valuable [2]. Whether an artifact is valuable is subjective. For the purposes of this paper, we will consider the minimal requirements for a story artifact to be considered valuable if it (a) meets the intended purpose of its creation and (b) is sufficiently mimetic – appearing to resemble reality, but in a way that it is more aesthetically pleasing than reality. In brief, stories should be novel, but not so novel that they are unrecognizable [3].

In this paper, we describe recent work on planning-based narrative generation. Planning is one model of narrative creation that has been shown to be favorable for story generation (c.f. [4], [5], [6], and [7; 8]). We are considering a model of interactivity in which a user is afforded the ability to specify a set of parameters that abstractly define a desired story. The system then responds to the request by generating a novel story that best meets the given parameters. However, other models of interactivity can also be considered such as *interactive stories*, where a user can participate in a story in real-time. See [9], [10] for techniques for creating interactive experiences by that recursively invoking plan-based narrative generators, and [1] for alternative approach to using planning in interactive narrative system. There are non-planning based approaches to interactive story systems – a non-exhaustive list includes [12], [13], [14], and [15] and derivate works – which are more dependent on hand-authored narrative content and therefore less applicable to the problem of scaling up the pace of content creation. However, further discussion of real-time interactivity is beyond the scope of this paper.

Our goal is to incrementally improve the ability of planning-based story generation by increasing the space of narratives that can be explored algorithmically. We believe that this will provide a capability to address the issue of consumption versus production, and also to create more customized experiences for users under either model of interactivity. While the problem of determining whether a generated story is good is still largely an open problem, we can make claims about the existence of stories with certain properties – properties that could not previously reliably be generated. In particular, we describe the vignette-based partial-order causal link (VB-POCL) narrative planning algorithm. VB-POCL implements a form of case-based planning that can incorporate vignettes – plot fragments that are a priori known to be "good" – into the narrative planning process.

2 Planning Stories

We view story generation as a problem-solving activity where the problem is to create an artifact – a narrative – that achieves particular desired effects on an audience. We favor a general approach where we model the story generation process as planning. One reason for this is that plans are reasonable models of narrative [6]. But also planners "walk the space" of possible narratives in search of a solution that meets certain qualities, making it a good model of creativity in general [2]. This follows from other research efforts modeling story generation as planning (c.f., [5], [7;8]).

The planning process is as follows: a planner chooses an incomplete plan to work on at the fringe of the problem space, and chooses a flaw in that plan to work on, resulting in zero or more new plans in which the flaw is repaired (and often

introducing new flaws). These plans become part of the new fringe, and the process is repeated. One type of flaw pertinent to this work is an *open condition flaw*, which exists when an action in the plan (or the goal state) has a precondition that has not been recognized as being satisfied by the effect of a preceding action (or the initial state). Applying one of the following repair strategies can repair the flaw:

(i) Selecting an existing action in the plan that has an effect that unifies with the precondition in question.
(ii) Selecting and instantiating an operator from the domain operator library that has an effect that unifies with the precondition in question.

A planner is non-deterministic, meaning it applies all strategies and then uses a heuristic function to determine which parts of the fringe should be expanded next. There are other types of flaws as well, such as causal threat flaws, which occur when an action threatens to undo the satisfaction of another action's preconditions. Conventional planners assure that plans are sound, meaning that they are guaranteed to execute successfully in the absence of unanticipated changes in the world [16].

However, stories are much more than just ways of achieving an intended outcome in the most efficient manner. Stories should meet the expectations of the audience. This may mean putting in details that are aesthetically pleasing even if they are not strictly necessary. When humans write stories, they call on their lifetime of experiences as a member of culture and society. A computer system that generates stories does not have access to this wealth of information. As a way of mitigating this handicap, a computer system can be provided with a wealth of knowledge in the form of traces of previous problem-solving activities or a library of previous solutions – in this case stories. *Transformational multi-reuse planning* is a form of case-based reasoning in which prior solutions are adapted to new planning problems; systems such as [17] and [18] retrieve and reuse full or portions of old solutions (e.g. plans) to assemble new plans. We adapt the transformational multi-reuse approach and customize it to the particulars of generating stories. However, instead of assuming a knowledge base of complete stories, we bootstrap the planning process with a library of "vignettes" – fragments of stories that capture some particular context.

2.1 Vignettes

We use the term vignette to refer to a fragment of a story that represents a "good" example of a situation and/or context that commonly occurs in stories [19]. For example, a library of vignettes would contain one or more specific instances of bank robberies, betrayals, cons, combat situations, etc. We do not presume to know how these vignettes were created, only that we have the solutions and that they have favorable mimetic qualities. It is important to note that the library contains specific examples of these situations instead of general templates. The implication of the existence of this library is that a story generator does not need to "reinvent the wheel" and thus does not need the specialized knowledge required to be able to create specialized narrative situations. Vignettes are fragments of story structure. How does one know what actions should be included in the vignette and which can be left out? We use the minimal vignette rubric: a *minimal vignette* is one in which removing any

```
Vignette:
Steps: 1: Start-Battle (?c1, ?c2, ?place)
       2: Wound (?c1, ?c2)
       3: Wound (?c1, ?c2)
       4: Mortally-Wound (?c2, ?c1)
       5: Die (?c1)
       6: End-Battle (?c1, ?c2)
Causation: 1→(battling ?c1 ?c2)→2
           1→(battling ?c1 ?c2)→3
           1→(battling ?c1 ?c2)→4
           1→(battling ?c1 ?c2)→6
           4→(mortally-wounded ?c1)→5
Constraints: (character ?c1)
             (character ?c2)
             (stronger ?c2 ?c1)
Variable-constraints: ?c1 ≠ ?c2
Ordering: 1→2, 2→3, 3→4, 4→5, 4→6
Effects: (battling ?c1 ?c2)
         (not (battling ?c1 ?c2))
         (wounded ?c2)
         (mortally-wounded ?c1)
         (not (alive ?c1))
```

Fig 1. An example vignette data structure.

one action from the vignette causes it to no longer be considered a good example of the situation and/or context it was meant to represent.

Computationally, vignettes are stored as plan fragments. As a plan fragment, it is possible that some actions do not have to have all of its preconditions satisfied. This is a way of saying that it is not important how the situation is established or even why, but once the conditions are established certain things should happen. Vignette plan fragments do not reference specific characters, objects, or entities so that a planner can fit the vignette into new story contexts by making appropriate assignments. To ensure illegal or non-sense assignments are not made, co-designation and non-co-designation variable constraints are maintained. Fig. 1 shows an example vignette capturing a very simple combat between two characters where one character (represented by the variable ?c2) is stronger than the other (represented by the variable ?c1). The weaker character wounds the stronger character twice before the stronger character delivers a mortally wounding blow. Finally, the mortally wounded character dies of its wounds. This vignette could be used in any plan in which a character must become wounded, mortally wounded, or dead, or plans in which battles must be started.

2.2 Planning Stories with Vignettes

The Vignette-Based Partial Order Causal Link (VB-POCL) planner is a modification of standard partial order planners to take advantage of the existence of a knowledgebase of vignettes. The VB-POCL planning algorithm is similar to other case-based planners such as [17] and [18] in that it adds a third strategy for repairing open condition flaws:

(iii) Retrieve and reuse a case that has an action with an effect that unifies with the precondition in question.

Given an action in the plan that has an unsatisfied precondition VB-POCL non-deterministically chooses one of the three above strategies. Strategies (i) and (ii) are performed in the standard way [16]. If strategy (iii) is selected, VB-POCL doesn't immediately repair the flaw. Instead, the plan is annotated with a fit flaw, indicating the plan is to be considered flawed until all actions from the vignette are fitted into the plan. Repairing a fit flaw is a process of selecting an action from the retrieved vignette and adding it to the new plan (or selecting an existing action in the plan that is identical to the selected action to avoid unnecessary action repetition). It may take

VB-POCL ($P, F, \Lambda_a, \Lambda_v$)

The VB-POCL algorithm takes a plan that is a partial solution to the problem (or the empty plan) P, a set of flaws F evidencing why P cannot be a solution, a library of un-instantiated operators Λ_a that represent templates of actions that characters can take in the world, and a library of vignettes Λ_v.

I. **Termination:** if $F = \emptyset$ and P is sound, return P. Otherwise, fail.

II. **Planning:**

A. **Goal selection:** Select an open condition flaw $f = <s_{need}, p_{need}>$ or a fit flaw $f = <P_c, s_c, e_c, p_{need}, s_{need}>$ from F. Let $F' = F \setminus \{f\}$.

B. **Operator selection:** Non-deterministically do one of the following (if valid):

1. Causal planning (if f is an open condition flaw): As normal, non-deterministically choosing and instantiating an action from Λ_a or reusing an instantiated operator already in P. If instantiating a new action, add open condition flaws for every precondition of the new action.

2. Vignette reuse (if f is an open condition flaw): Non-deterministically retrieve a plan fragment P_c from the vignette library Λ_v such that some step s_c in P_c has an effect e_c that unifies with p. Let f' be a fit flaw such that $f' = <P_c, s_c, e_c, p_{need}, s_{need}>$. $F' = F' \cup \{f'\}$. Let P' be a copy of P.

3. Plan refitting (if f is a fit flaw): Choose a step s_{next} from P_c that hasn't been chosen before. Let P' be a copy of P. Let s_{add} be s_{next} or a step in P' that is identical to s_{next} (choose non-deterministically). If $s_{add} = s_{next}$, add s_{add} to P', including relevant causal links, temporal links, and variable bindings. Else, only add relevant causal links, temporal links, and variable bindings from P_c. If $s_{next} = s_c$ then add a causal link from s_{add} to s_{need} in P'. Add necessary open condition flaws to F' for s_{add} for each precondition that will not be satisfied by a causal link in P_c. Let $f' = <P_c', s_c, e_c, p_{need}, s_{need}>$ where P_c' is a copy of P_c with s_{next} removed. $F' = F' \cup \{f'\}$.

C. **Threat resolution:** Performed in the standard way (if no resolution exists, backtrack).

III. **Recursive Invocation:** Call VB-POCL($P', F', \Lambda_a, \Lambda_v$).

Fig. 2. The vignette-based planning algorithm.

several invocations of the fitting procedure to completely repair a fit flaw. This may seem more inefficient than just adding all vignette actions to the plan at once. However, there are three advantages to iterative fitting. First, it is easier to recognize and avoid action repetition. Second, it allows for interleaving of repair of other flaws, which can lead to discovery of interesting plans. For example, fitting may lead to the creation of new open condition flaws that in turn are repaired through conventional planning (strategies i and ii) or by retrieving new vignettes (strategy iii). Third, problems in the fitting process can be identified sooner in case the strategy must be abandoned.

The algorithm for VB-POCL is given in Fig. 2. VB-POCL is instantiated with an empty plan P, a set of flaws F, and libraries of un-instantiated action templates and vignettes. Initially, P is an empty plan that only contains information about the initial state – the description of the story world before the story begins – and the outcome state – the description of what the human user wants the story world to be like at the end of the story. VB-POCL selects a flaw. Initially the only flaws are that the outcome state is made up of state propositions that are unsatisfied. As described earlier, open condition flaws, in which an action's precondition (or an outcome state proposition) is unsatisfied, are repaired by three strategies. Strategies (i) and (ii) make up conventional planning (c.f. [16]) and are represented as *causal planning* in Fig. 2. Strategy (iii) is initially handled by the *vignette reuse* portion of the algorithm in Fig. 2. VB-POCL retrieves all vignettes that can satisfy the open condition. How

this retrieval happens is relatively simple, but is beyond the scope of this paper. Each successful retrieval creates a branch in the problem space. For each branch, a fit flaw is created, storing the vignette (P_c), information about the action and precondition that is not satisfied (s_{need} and p), and information about which action in the vignette – called the *satisfier action* – can be used to satisfy the original open condition flaw (s_c and e_c). A vignette can be retrieved multiple times with different satisfier actions. The algorithm resolves any causal threats in the normal way (c.f. [16]), and iterates.

When VB-POCL chooses to work on a fit flaw, the *plan refitting* portion of the algorithm is invoked. An action is arbitrarily selected from the vignette and instantiated into the plan (or an identical action that already exists in the plan is chosen). The order doesn't affect the soundness or completeness of the algorithm. All necessary causal links, temporal links, and variable bindings are added to the plan to ensure proper placement and character references of the new or existing action. A special case occurs when the action selected from the vignette is the action that should be used to satisfy the original open condition flaw on action s_c. When this happens, an extra causal link is extended from the vignette action to the original action with the unsatisfied precondition. This finally solves the open condition flaw that prompted the vignette retrieval in the first place. To complete the plan refitting process, the action selected from the vignette is removed from the copy of the vignette P_c, and the new plan is annotated with a new fit flaw that has a slightly smaller vignette. The algorithm resolves any causal threats and iterates.

Planning is complete when a plan is found on the fringe that has no flaws.

2.3 Example

To illustrate the VB-POCL planning algorithm, we provide an example of how the planner could use the vignette shown in Fig. 1. Suppose we wanted a story set in J.R.R. Tolkein's Middle Earth. The story world is in the state in which one character, called Enemy, has in his possession a Silmiril – a precious magical stone. The outcome, provided by the human user, is that another character, called Hero, gains possession of the Silmiril. In the remainder of this section, we trace the planning process, describing only one of many possible paths that VB-POCL can follow[1]. The planner starts by non-deterministically choosing to satisfy the goal by having Hero take the Silmiril from Enemy. This requires that the Enemy not be alive. The planner could use the vignette from Fig. 1 here by retrieving it and binding Enemy to *?c1*. Note that this strategy will eventually fail because it would require a character stronger than Enemy. Instead the planner chooses to use conventional planning to instantiate an action, *Die-of-Infection*, that causes Enemy to not be alive. This requires that Enemy be superficially wounded. Here VB-POCL retrieves the vignette from Fig. 1 because it has an action that can have the effect (once variables are bound) of causing Enemy to become wounded. Each vignette action is spliced into

[1] The VB-POCL algorithm follows all possible paths to solving a problem in a best-first manner. We use the term *non-deterministic* to gloss over all the incorrect choices at any given decision-point until it makes a choice that leads to a solution.

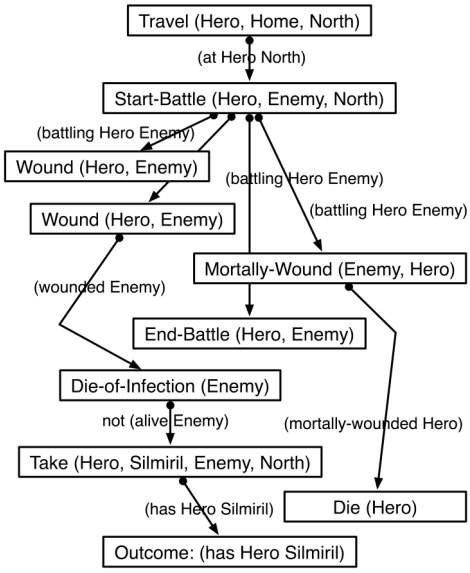

Fig 3. A story plan generated by VB-POCL.

the new story plan one at a time, using the process of refitting described earlier. Determining where in the plan to splice an action is resolved by repairing causal and temporal inconsistencies (e.g. causal threat flaws). For example, when *Die(Hero)* is spliced into the story plan, it must be temporally ordered after *Take* to avoid inconsistencies; for a character to *Take* an item, the character cannot be dead.

The vignette is fairly self-contained, but the vignette action, *Start-Battle* does require that the planner establish that both Hero and Enemy are at the same place, which in this case the North. This precondition is satisfied in the normal way, by using conventional planning strategies to instantiate an action in which Hero travels to the North (Enemy is already there). The final story plan is shown in Fig. 3. Boxes are actions and arrows represent causal links. A causal link indicates how an effect of one step establishes a world state condition necessary for a precondition of latter steps to be met. For clarity, only some preconditions and causal links on each action are shown.

2.4 Vignette Transformation

VB-POCL assumes a library of vignettes that are already in the domain of the story to be generated. A domain is a set of propositions that describe the world, including characters, and a set of operator templates that described what characters can do and ways in which the world can be changed. In the example, the domain describes characters such as Hero and Enemy and operators such as *Travel* and *Wound*. However, we may want the story planner to have access to vignettes from other domains, especially if our new story is set in an unique and specialized story world

domain. We use the term *far transfer* to refer to the process of transferring a vignette from one domain to another.

To engage in far transfer on vignettes, one must first find analogies between domains. Analogy-finding algorithms have been demonstrated to be able to find analogies between stories when stories are pre-existing. Far transfer differs from the problem of finding analogies between stories because there is not a second instance of a story to compare. Instead, we search for analogies between domains and use that information to translate a known vignette from one domain to another.

The far transfer process is summarized as follows. A *source vignette* is a vignette in an arbitrary domain, called the *source domain*. The *target domain* is the domain of the story to be generated. For each action in the source vignette, the far transfer algorithm searches for an action in the target domain that is most analogical. The search involves a single-elimination tournament where target domain actions compete to be the most analogical according to the Connectionist Analogy Builder (CAB) [20]. The winner of the tournament is the target domain vignette most analogical to the source domain vignette. The result is a mapping of source domain actions to target domain actions that can be used to translate a source vignette into a target domain through substitution.

The far transfer algorithm runs CAB $m*n$ times, where m is the number of actions in the source domain vignette and n is the number of actions in the target domain. The algorithm's complexity is subsumed by the complexity of CAB, which is NP-complete.

Translated vignettes may have gaps where translation is not perfect. This is not a problem because the VB-POCL will recognize this and fill in the gaps via planning. Applying this process to all vignettes in a library results in a new library in which all vignettes are in the proper domain. See [19] for a detailed description of far transfer.

2.5 Discussion

One of the interesting properties of VB-POCL is that vignette retrieval can result in story plans in which there are actions that are not causally relevant to the outcome. Trabasso and van den Broek [21] refers to actions that are causally irrelevant to the outcome as dead-ends. In the example above, the causal chain involving Enemy mortally wounding Hero and then Hero dying appears to be a dead-end because those actions do not contribute to Hero acquiring the Silmiril. Dead-ends are not remembered as well as actions that are causally relevant to the outcome [21], suggesting that dead-ends should be avoided. A battle in which a single wound was inflicted on Enemy would have sufficed, and this is what planners such as [16] and [7] would have settled on.

Human authors regularly include dead-end events in stories suggesting some importance to dead-ends. We hypothesize that there are certain mimetic requirements to be met in any story and that dead-ends can serve this purpose. For example, we assume that a combat scenario in which many blows of varying strengths are exchanged is more interesting than a combat in which a single blow is dealt. Interestingly, what may be a dead-end causal chain to the story planner may not be considered a dead-end by a human reader, and vice versa. That is, the reader may

interpret the example story as a tragedy and consider the death of Hero as one of two primary causal chains, whereas the planner's representation contains only one causal chain that leads to the human user's imposed outcome (Hero has the Silmiril). More research needs to be done to create intelligent heuristics to recognize when dead-ends (from the planner's perspective) are favorable, tolerable, or damaging.

VB-POCL is capable of finding stories that other causal-planning based story generation techniques are not able to find. Specifically, these are stories in which some actions are not strictly necessary for causal achievement of some human-specified outcome state. As noted in [8], expanding the space that can be explored provides an opportunity to find more solutions that are valuable. However, one could claim that some – or all – of the creativity occurred in the process of transforming vignettes, executed prior to generation. As a first step toward improving the ability of planning-based story generation to reliably produce valuable, mimetic stories, the VB-POCL algorithm provides the technical capability of searching a large space of possible solutions. Future work requires strategies for controlling the search space exploration, including heuristics for ranking solution "goodness." That is, VB-POCL currently has no understanding of how multiple vignettes add or detract from each other or the overall quality of the story being generated.

On a practical note, planning stories with vignettes is a way to increase the average length of stories that can be generated. Ideally, a planner should only have to make O(n) decisions where n is the length of the plan generated. In practice, planners backtrack, meaning that they spend time generating action sequences that do not pan out and must return to an earlier decision point. Any effort spent on a line of reasoning that does not pan out is wasted effort. In the worst case, a planner must consider all ways of making every decision (O(b^n) where b is the number of ways a decision can be made, and n is the length of the solution [16]). Vignettes, when selected, guide the process of adding actions to the story plan, offering up actions in hand-coded sequences that are less likely to result in backtracking than if every action must be chosen independently. Of course, VB-POCL can interleave multiple vignettes during which time new issues that cause backtracking can arise; this is the price of flexibility. Future work is needed to develop powerful heuristic functions that can help VB-POCL discriminate between vignettes when more than one can be applied to an open condition flaw. The practical result of less backtracking is that more time can be spent of fruitful action sequences, potentially allowing for longer plans to be created in less time.

3 Related Work

Search based narrative generation approaches include Tale-Spin [22], which uses a simulation-like approach, modeling the goals of story world characters and applying inference to determine what characters should do. Dehn [4] argues that a story generation system should satisfy the goals of the human user. That is, what outcome does the user want to see? The Universe system [5] uses means-ends planning to generate an episode of a story that achieves a user's desired outcome for the episode. More recent work on narrative generation attempts to balance between character goals

and human user goals [7]. Further work on story planning addresses expanding the space of stories that could be searched [8].

Case-based reasoning (c.f. [23]) has been found to be related to creativity [2; 24]. Several approaches to narrative generation use case-based reasoning. Minstrel [25] implements a model of cognitive creativity based on routines for transforming old stories into new stories in new domains. ProtoPropp [26] uses case-based reasoning to generate novel folk tales from an ontological case base of existing Proppian stories. Mexica [27] uses examples of prior stories to propose plot points and then applies means-ends planning to fill in missing details. VB-POCL is an attempt to harness the power of search-based generation and case-based creativity in a formalized causal planning framework.

VB-POCL is a variation on case-based reasoning. Case-based reasoners typically engage in four processes: *retrieve*, *reuse*, *revise*, and *retain* [23]. Transformational multi-reuse planners attempt to reuse components of solutions to similar problems to solve new problems, thus possibly invoking retrieve, reuse, and revise processes more than once. VB-POCL is most similar to [17] and [18], but differs from them and all other case-based planners in the following ways. First, vignettes are not complete solutions to previously solved problems; a vignette is not a case. But vignettes are used like a case. Regarding VB-POCL functionality, the VB-POCL retrieval process retrieves all vignettes that can conceivably be used to satisfy an open condition. Typically, the cost of retrieval and reuse is very high so a system must deliberate about the cost tradeoff of standard planning versus retrieval and reuse. Trying all vignettes that meet the requirements for retrieval is not practical if vignettes require extensive modifications for reuse. VB-POCL assumes that vignettes in the library are in the domain of the story being generated and thus do not require extensive effort for reuse. An offline algorithm – summarized in Section 2.4 and described in detail in [19] – is used to transform all vignettes from arbitrary domains into the domain of the new story to be generated. That is, many of the computationally intensive aspects of *reuse* occurs offline and thus fitting a vignette into a plan is trivial. Second, vignettes don't require modification because vignettes are *minimal*. Reuse is performed by blindly inserting all actions in a retrieved vignette into the plan; VB-POCL does not need to make hard decisions about which actions should be kept and which actions should be discarded. Finally, because vignettes are not cases, VB-POCL does not reincorporate (e.g. retain) its solution story plans back into the knowledge base. That is, VB-POCL does not attempt to learn to solve problems from past examples. One of the interesting properties of transformational multi-reuse planning algorithms such as [17] and [18] is that they can operate when there are no applicable cases available; the algorithm can fall back on conventional planning. VB-POCL shares this trait, but unlike transformational multi-reuse planners VB-POCL is also complete, meaning it can find all solutions that exist (the proof is beyond the scope of the paper).

It may be possible to use hierarchical task network (HTN) planners [28] or a decompositional planner such as DPOCL [29] to achieve similar effects as VB-POCL. However, using HTNs or other decompositional techniques to generate story requires reasoning at higher levels of abstraction than the action (or event), and this introduces potentially rigid top-down structuring of plot that can limit opportunistic discovery such as in [7; 8]. Further, vignettes can potentially come from many sources, which may or may not be accompanied by abstract context information.

There are many similarities between VB-POCL and macro-operator planners. Indeed, VB-POCL's vignette retrieval process can be considered analogous to selecting a virtual macro-operator. Macro-operator planners transform an action space into a more compact action space for efficiency gains by learning to group primitive actions that occur together frequently into abstract operators [30]. However, VB-POCL doesn't learn vignettes. Indeed, vignettes often contain action sequences that *cannot* be found by the planner. Further, VB-POCL needs to operate in the primitive action space so that vignettes can be spliced together or so that VB-POCL can use conventional planning techniques to discover new action sequences.

4 Conclusions

VB-POCL is a planning algorithm that extends conventional planning algorithms (e.g. [16]) to make it more applicable to narrative creation. Specifically, VB-POCL extends the conventional planning algorithm by retrieving and reusing vignettes. This is a strategy for tapping into the experiences of other presumably expert story authors. VB-POCL shares many similarities with case-based planners such as [17] and especially [18]. However, by treating plans as narratives – that is, the plan is not a schedule of actions to be executed for goal attainment but a description of events that lead to an outcome – we are able to simplify the case (vignette) reuse problem by assuming that our library of vignettes includes only minimal vignettes.

VB-POCL can explore a greater space of stories because it can consider story plans that have action that are not causally necessary to reach some given outcome. We believe that some of these stories will be more valuable because of the mimetic qualities of the vignettes and the potential for these stories to possess both global novelty and localized familiarity. While we have not yet performed an evaluation of VB-POCL, we believe that this is a step towards instilling computer systems with the ability to assume responsibility for story content creation. This can be an important for application areas where content creation is a bottleneck and it is possible for the pace of content consumption to overtake the pace of content production.

References

1. Morningstar, C. and Farmer, F.R.: The Lessons of Lucasfilm's Habitat. In: M. Benedikt (Ed.) *Cyberspace: First Steps*. MIT Press (1991).
2. Boden, M.: *The Creative Mind: Myths and Mechanisms, 2nd Edition*. Routledge (2004).
3. Giora, R.: *On Our Mind: Salience, Context, and Figurative Language*. Oxford University Press (2003).
4. Dehn, N.: Story Generation after Tale-Spin. In: *7th International Joint Conference on Artificial Intelligence* (1981).
5. Lebowitz, M.: Story-Telling as Planning and Learning. *Poetics*, 14, 483—502 (1985).
6. Young, R.M.: Notes on the Use of Plan Structures in the Creation of Interactive Plot. In: *AAAI Fall Symposium on Narrative Intelligence* (1999).
7. Riedl, M.O., Young, R.M.: An Intent-Driven Planner for Multi-Agent Story Generation. In: *3rd International Joint Conf. on Autonomous Agents and Multi Agent Systems* (2004).

8. Riedl, M.O., Young, R.M.: Story Planning as Exploratory Creativity: Techniques for Expanding the Narrative Search Space. *New Generation Computing*, 24(3), 303—323 (2006).
9. Young, R.M., Riedl, M.O., Branly, M., Jhala, A., Martin, R.J., Saretto, C.J.: An Architecture for Integrating Plan-Based Behavior Generation with Interactive Game Environments. *Journal of Game Development,* 1, 50—70 (2004).
10. Riedl, M.O., Stern, A., Dini, D., Alderman, J.: Dynamic Experience Management in virtual Worlds for Entertainment, Education, and Training. *International Transactions on Systems Science and Applications*, 4 (2008).
11. Barber, H.M., Kudenko, D: Dynamic Generation of Dilemma-based Interactive Narratives. In: *3rd AI and Interactive Digital Entertainment Conference* (2007).
12. Magerko, B.: Evaluating Preemptive Story Direction in the Interactive Drama Architecture. *Journal of Game Development*, 2 (2007).
13. Mateas. M., Stern, A.: *Architecture, Authorial Idioms and Early Observations of the Interactive Drama Façade*, Technical Report, CMU-CS-02-198, School of Computer Science, Carnegie Mellon University (2002).
14. Si, M, Marsella, S., Pynadath, D.V.: Thespian: Using Multi-Agent Fitting to Craft Interactive Drama. In: *4th International Joint Conference on Autonomous Agents and Multi Agent Systems* (2005).
15. Weyhrauch, P.: Guiding Interactive Fiction. Ph.D. Disseration, Carnegie Mellon University (1997).
16. Weld, D.: An Introduction to Least Commitment Planning. *AI Magazine*, 15(4), 27-61 (1994).
17. Francis, A.G., Ram, A.: A Domain-Independent Algorithm for Multi-Plan Adaptation and Merging in Least-Commitment Planners. In: *AAAI Fall Symposium on Adaptation of Knowledge for Reuse* (1995).
18. Britanik, J., Marefat, M.: CBPOP: A Domain-Independent Multi-Case Reuse Planner. *Computational Intelligence*, 20, (2004).
19. Riedl, M.O., León, C.: Toward Vignette-Based Story Generation for Drama Management Systems. In: *Workshop on Integrating Technologies for Interactive Story* (2008).
20. Larkey, L.B., Love, B.C.: CAB: Connectionist Analogy Builder. *Cognitive Science,* 27, 781—794 (2003).
21. Trabasso, T., van den Broek, P.: Causal Thinking and the Representation of Narrative Events. *Journal of Memory and Language*, 24, 612-630 (1985).
22. Meehan, J.: *The Metanovel: Writing Stories by Computer*. Ph.D. Dissertation, Yale University (1976).
23. Kolodner, J.: *Case-Based Reasoning*. Morgan Kaufmann Publishers (1993).
24. Kolodner, J.: Understanding Creativity: A Case-Based Approach. In: *1st European Workshop on Case-Based Reasoning* (1993).
25. Turner, S.: *MINSTREL: A computer model of creativity and storytelling*. Ph.D. dissertation, University of California , Los Angeles (1992).
26. Gervás, P., Díaz-Agudo, B., Peinado, F., Hervás, R. Story Plot Generation Based on CBR. *Journal of Knowledge-Based Systems*, 18(4-5), 235-242 (2006).
27. Pérez y Pérez, R., Sharples, M.: Mexica: A Computer Model of a Cognitive Account of Creative Writing. *Journal of Experimental and Theoretical Artificial Intelligence*, 13(2), 119—139 (2001).
28. Sacerdoti, E.D.: A Structure for Plans and Behavior. Elsevier, New York (1977).
29. Young, R.M., Pollack, M., Moore, J.: Decomposition and Causality in Partial-Order Planning. In: *2nd International Conference on AI and Planning Systems* (1994).
30. Gratch, J., DeJong, G.: A Framework of Simplifications in Learning to Plan. In: *1st International Conference on Artificial Intelligence Planning Systems* (1992).