User-Driven Narrative Variation in Large Story Domains using Monte Carlo Tree Search

Bilal Kartal, John Koenig, and Stephen J. Guy
Department of Computer Science
University of Minnesota
Minneapolis, MN, 55455
{bilal,koenig,sjguy}@cs.umn.edu

ABSTRACT

Planning-based techniques are powerful tools for automated narrative generation, however, as the planning domain grows in the number of possible actions traditional planning techniques suffer from a combinatorial explosion. In this work, we apply Monte Carlo Tree Search to goal-driven narrative generation. We demonstrate our approach to have an order of magnitude improvement in performance over traditional search techniques when planning over large story domains. Additionally, we propose a Bayesian story evaluation method to guide the planning towards believable narratives which achieve user-defined goals. Finally, we present an interactive user interface which enables users of our framework to modify the believability of different actions, resulting in greater narrative variety.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Multiagent systems

Keywords

Monte Carlo Tree Search; Upper Confidence Bounds; UCB; MCTS; Exploration versus Exploitation

1. INTRODUCTION

Computer generated narratives can be important for many types of immersive virtual environments, with potential applications in diverse areas such as training, education, and entertainment. For example, computer games could ideally create new plots for player characters on each play through. Likewise, virtual libraries of computer generated books could appear in games, each with their own unique stories. Ultimately, automatic narrative technologies can lead to author assistance tools and simple stories for practicing reading skills.

While there has been great progress in many aspects of interactive media, the ability to automatically generate narratives has not similarly improved. This is due in part to inherent challenges of natural language processing, but also difficulties in producing believable and artistic narratives from realistic, interesting story domains. In this paper, we focus on the latter, generating stories where the actions of multiple characters believably come together to achieve interesting, user-defined goals. At a high-level, this problem can be well formulated as a planning problem. However, if there are large number of possible actors, actions, and places there is an exponential explosion in the search space being planned over.

To overcome this difficulty, we propose leveraging the recently developed planning technique called Monte Carlo Tree Search (MCTS). MCTS has shown promising results in several domains with large search spaces and has been a game changer for several AI problems [8]. One of the most notable examples is computer Go, where computers have shown the ability to beat top human professionals (on a 9x9 board) [9] despite Go's large search space. Inspired by this success, we exploit MCTS within the context of narrative generation from rich domains with many actors, places, etc..

Rather than just finding a single narrative which meets the given user-defined goals, we are interested in being able to generate a variety of different stories which meet narrative goals in a *believable* fashion. To this end, we introduce a believability metric which allows users to guide MCTS as it plans within a given story domain. Our formulation of this metric assigns a contextually sensitive believability score to each possible action in the story domain. By varying the believability of actions, we allow a high degree of user-controlled variation in the generated narratives.

Main Results: Our main contributions are as follows:

- A new algorithm for computer narrative generation based on MCTS and evaluation of the effect of various search heuristics in terms of performance and memory usage.
- A novel metric based on contextual believability to guide the planning processing efficiently towards stories which believably reach user-defined goals.
- A graphical-interface which allows user-driven variation in the generated narratives.

Organization The rest of this paper is organized as follows. Section 2 highlights related work and Section 3 presents a high-level overview of our approach to the narrative generation problem. In Section 4, we present results from our system along with their analysis, and in Section 5 we discuss the resulting interactive framework for user-driven narrative variation. Finally, Section 6 concludes the paper with limitations of our approach and future work.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*
Copyright © 2014, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2. PREVIOUS WORK

In this section we briefly discuss previous work in the area of automated narrative generation and highlight some related work which makes use of Monte Carlo Tree Search for different problem domains.

2.1 Automated Narrative Generation

Narrative generation problem has been studied with an increased focus within the last decade. We will present different approaches and also discuss related work similar to our approach in literature.

2.1.1 Data-driven approach

Data-driven techniques are employed to exploit story corpus to either learn important parameters to be used for planning phase or to improve planner's decisions with similar cases during narrative generation. For example, McIntyre et al. [14] presented a story generation system that uses a database to automatically infer a story domain which would otherwise have to be manually entered by a user. Gervas et al. [11] proposes a case-based system for story generation process. Their framework has a sample story database and a given a new story query, they generate the overall structure of story plots with case comparisons.

2.1.2 Character-centric approach

Character-centric approaches deal with narrative generation by granting some of the narrative characters the role of story level design so that these characters ease the global planning process. For example, the work of Theune et al. [29] on Virtual Storyteller models autonomous agents and assigns them roles within the story by an external plot-agent. Brenner [3] studied the narrative generation problem within the domain of multi-agent planning. More recently, Teutenberg et al. [28] combined intentional planning with the multi-agent planning of Brenner [3].

2.1.3 Interactive systems

Skorupski et al. [25] proposed a very detailed interactive framework as an author tool that enables the user to interact with the ongoing story either by being a third person character or acting on behalf of any AI character during the generation process. The work proposed by Barber et al. [2] presents an interactive framework named *GADIN* aiming to generate narratives that have dramatic tensions. Their framework works with a global story goal, and they studied children story domains. Lastly, Cavazza et al. [5] proposed a character based interactive story telling system where the user interacts any time with the ongoing story by either controlling an agent or speaking with the story agents through a speech recognition system.

A more recent work on interactive narrative generation is presented in [23] which proposes a real-time event centric story generator that can handle both cooperative and adversary user interactions in a flexible way so that given goals are still satisfied. Lastly, Permar et al. [17] presented a computational model for cognitive script generation for interactive narratives.

2.1.4 Other approaches

Author-centric methods such as analogy-based story generation of SAM [15] and MEXICA [16] attempt to generate narratives from the point-of-view of the author. Riedl et al. [19] presented a story generation scheme that considers both story plot coherence and intentions of story characters during planning. Swartjes et al. [27] proposed a simulation based formulation to generate narratives in an emergent manner. Riedl et al. [20] also proposed a novel approach to evaluate believability of computer generated narratives by establishing the causal relationship with actions and characters' intention and perception of the story world. Lastly, story-centric methods, such as Fabulist [21], reason over intentions and corresponding actions from the point of view of the audience. In doing so, story-centric methods generate narratives which more clearly communicate character motivations to the audience.

2.2 Multi-Agent Planning for Narrative Generation

There are several deterministic planners which can be employed for the narrative generation problem. For Example, Richter et al. [18] proposed an anytime planner, LAMA. Helmert et al. [12] presented the planner named Fast Downward Stone Soup which uses several heuristics and different planner sequentially to accomplish the search. Other approaches includes LPG-d [26] which can create a set of solutions specified by the parameter d for a given problem.

2.2.1 Monte Carlo Tree Search

Monte Carlo Tree Search is a powerful method, that has shown particular success in searching over large domains by using random sampling approaches. It is an anytime algorithm that will improve its solutions provided more time and memory. Its success has been particularly recognized after its performance in the game Go [9]. MCTS has been further employed in real-time games [22] and Solitaire puzzles [6] where it can outperform humans. MCTS has also been used for other domains including optimization [24] and planning problems [7]. For more information, we refer reader to the excellent survey presented in [4]. We employed MCTS with UCB (Upper Confidence Bounds) following the approach from [13] which balances exploration vs. exploitation during planning.

3. MCTS FOR STORY GENERATION

In this section, we first introduce a new story domain in which we evaluate our method. We then introduce our believability metric that guides the MCTS search, and provide a detailed explanation of our planning method.

3.1 Story Domain

Planning based story generation typically works over a user-specified story domain. We support a custom domain based on a simplified PDDL-type [10] of environment specification. While our approach is generic, we demonstrate it using the following crime-story inspired domain.

Our domain has three types of entities: *Actors*, *Items*, and *Places*. *Actors*, which are intended to represent people or other characters, can pick up or use various *Items*, or move to other places. Each *Item* allows different actions for an *Actor*. *Items* and *Actors* can be located at different *Places*.

Each entity has several *attributes* which allows the planner to keep track of what effect various actions have on the *Actors*, *Items*, and *Places*. For example, actors have a "health"

attribute which is decreased when they are attacked. Below is an abbreviated list of the various actions allowed in our story domain followed by a brief description of its affect:

- **Move(A, P):** A moves to place P.
- **Arrest(A, B):** B's place is set to jail.
- **Steal(A, B, I):** A takes item I from B. This increase B's anger.
- **Play Basketball(A, B):** A and B play basketball. This decreases A's and B's anger.
- **Kill(A, B):** B's health to zero (dead).
- **FindClues(A):** A searches for clues at its current location
- **ShareClues(A, B):** A shares with B any clues he has found.
- **Earthquake(P):** An earthquake strikes at place P. This causes people at P to die (heath = 0), items to be stuck, and place P to collapse.

Associated with each action are methods to convert the action and corresponding *Actors*, *Items*, and *Places* to English text.

For *Actors* we have several citizens: Alice, Bob, Charlie, David, etc. There is also a detective named Sherlock, and an inspector named Inspector Lestrade. For *Places* there are several homes, recreation areas (e.g., basketball courts), and a downtown. *Items* include flower vases, basketballs, baseball bats, guns and handcuffs. As discussed below, the believability of an actor taking a certain action will depend on where they are, what items they have, and their past experiences with other people.

We assume that the user specifies both an initial configuration and a goal for the story (e.g., who is in their own house, who is in downtown, where are the guns and vases). An example goal might be, "at least two people are dead and the murderer is arrested". For the purpose of running experiments, we can make the domain more complex by adding more citizens, items and places, and by changing the goal.

3.2 Believability

Our approach focuses on goal-oriented narrative generation. However, rather than searching to find any story which satisfies a user's goal we search for the best-possible story as evaluated by our metric. For this work, we chose a broad evaluation criteria based on how believable an action is contextually, given the current state of the story. The believability of each action is a user-defined measure on a scale from 0 to 1, which we treat as a Bayesian probability. That is, given the current state of the world, how likely it is that an event happens conditioned on the current state of the environment. For example, character A attacking character B may be more believable if A is angry. Likewise, a character arresting someone may be more believable if the character is an inspector. Some key examples from our domain are presented below.

- **Arrest(A, B)** More believable if A is an inspector. More believable if A has clues to a crime.
- **Steal(A, B, I)** More believable if item I is valuable.
- **Kill(A, B)** More believable if A is angry. More believable if A has previously killed someone.
- **FindClues(A, P)** More believable if A is an inspector or a detective.
- **ShareClues(A, B)** More believable if B is an inspector.
- **Earthquake(P)** Very low believability.

For a series of actions, we evaluate the overall believability as the product of the believability of each individual action:

$$B(a_1, a_2, ..., a_n) = \prod_{i=1}^{n} B_{a_i} \tag{1}$$

3.3 Approach Overview

Our approach uses the MCTS algorithm to find the chain of actions which accomplishes the user-defined goals with the maximum amount of believability. To apply MCTS we must first define a function which evaluates the extent that a given story believably reaches the user's goals. This function will shape the expansion of the Monte Carlo search tree.

Formally, we represent a given story as a set of actions $\mathcal{A} = \{a_1 \cdots a_n\}$. We define a story evaluation function E as:

$$E(\mathcal{A}) = G(\mathcal{A})B(\mathcal{A}) \tag{2}$$

where $G(\mathcal{A})$ is the percentage of the user-defined goals the current story accomplishes, and $B(\mathcal{A})$ is the believability of the story as specified in Eqn 1.

There is a tradeoff between overall believability of a generated story and the number of goals it achieves; a story that maximizes the value of $E(\mathcal{A})$ simply finds such an optimal tradeoff which does not necessarily completes all the goals.

Importantly, this formulation allows for a series of actions that are not very believable to occur in the story if it is the only way to achieve the user's specified goals.

While $E(\mathcal{A})$ provides a natural way to evaluate a completed story, it is of limited use for partial narratives that will be encountered during a tree search. This is because until a story satisfies some of the goals, the evaluation will always be 0. We address this issue by adding a random *rollout* to the story, that is a series of random actions that is added to the partial story until all the goals are met (or until a story grows past a length threshold). We denote this randomized extension of $\mathcal{A}$ as $\mathcal{A}'$:

$$\mathcal{A}' = \{a_1, a_2, ...a_n, r_1, r_2, ...r_n\}. \tag{3}$$

where $r_1 \cdots r_n$ are randomly generated actions. This allows a probabilistic evaluation of $\mathcal{A}$ even when $\mathcal{A}$ does no yet reach the goal. We denote this probabilistic evaluation as E':

$$E'(\mathcal{A}) = E(\mathcal{A}'). \tag{4}$$

We can now formulate story generation as a Monte Carlo tree search problem. Each node in the tree will represent the complete state of the world. Each link in the tree represents one possible action from that state, and that child of the node represents the resulting world state after applying that action. The root of the tree is the initial state of the world. The MCTS algorithm proceeds by repeatedly adding one node at a time to the current tree. For each potential action, we keep track of how many times we have tried that action, and what the average evaluation was. Choosing which child node to expand (i.e., choosing which action

to take), becomes an exploration/exploitation problem. We want to primarily choose actions that had good scores, but we also need to explore other possible actions in case the random rollout was "unlucky" and does not represent their true potential of that action.

The exploration/exploitation dilemma has been well studied in other areas. Here, we chose to use the Upper Confidence Bounds (UCB) approach proposed by [1]. This overall process of combining MCTS with UCB is often referred to as Upper Confidence Trees (UCT)[13]. Applied to our framework, this means that each node chooses its child n with the largest value of $f(n)$:

$$f(n) = E'(\mathcal{A}_n) + \sqrt{\frac{2\ln v}{n_v}} \tag{5}$$

where $\mathcal{A}_n$ is the parent's story so far updated to include action n, v is the total number of times this node has been visited, and n_v is the total number of times that given child action has been previously tried.

Choosing which node to add is then a recursive process. For each node, a child action with the largest value of the UCB equation (Eqn. 5) is chosen and expanded. When a node with unexplored child is reached ($n_v = 0$) a new node is created for one of the unexplored children. The process then starts again from the root of the tree, each time adding one new node. This way, the tree can grow in an uneven manner, biased towards nodes with high value for $E'(\mathcal{A}_n)$, which are likely to be good narratives. This process is summarized in Algorithm 1, the algorithm takes as input a *budget* of the maximum number of nodes to explore and returns a series of actions which comprise the story.

Algorithm 1: MCTS Story Generation

```
Input  : Budget
Output: Best story
while budget > 0 do
    Node ← ucbSelection(root) ;
    result ← rolloutStory(node) ;
    backpropagate(result) ;
    if result > bestScoreSoFar then
        updateBestScore();
        saveBestStory();
    end
end
return Best Story;
```

3.4 Iterative Implementation

The MCTS algorithm as presented in Algorithm 1 keeps the entire tree in memory and can exhaust memory when exploring domains with large branching factors (see Figure 5). This can be alleviated by pruning sections of the search tree that are unlikely to be productive. To this end, we propose an iterative approach which plans the story only one action at a time. This approach first grows the tree for a fixed number of actions. Then, only the current best action is kept, and its sibling actions' and their subtrees are pruned. This action forms the new initial condition and the tree search continues. Pseudocode for the iterative approach is presented in Algorithm 2.

As only a fixed number of nodes are added between each pruning step, the amount of memory used is bounded. We should note that this iterative approach is no longer probabilistically complete, as it is possible to prune a promising branch early on, leading to a local maxima rather than the global optimum. However, in practice we are still able to generate high scoring narratives while using much less memory than the non-iterative approach.

Algorithm 2: Iterative Story Generator

```
Input  : Budget and max_iterations
Output: Best story
for i ← 1 to max_iterations do
    while budget > 0 do
        Node ← uctSelection(root);
        result ← rolloutStory(node);
        backpropagate(result);
        if result > bestScoreSoFar then
            updateBestScore();
            saveBestStory();
        end
    end
    root ← root's most visited child;
    Prune all other subtrees;
end
return Best Story;
```

3.5 Search Heuristics

Monte Carlo Tree Search can be improved by applying heuristics to help guide the search. We incorporate two domain independent heuristics. For both heuristics, we keep a history table that stores average evaluation results, E', for each action (independent of it's depth in the tree). We explore two ways of using this history table: *selection biasing* and *rollout biasing*.

3.5.1 Selection Biasing

Here we modify Eqn. 5 to incorporate the average value for the action stored in the history table. We introduce a parameter α which weighs the history average value more strongly when very few (less than k) rollouts have been performed. Formally:

$$f(n) = \alpha E'(\mathcal{A}_n) + (1-\alpha)H(n) + \sqrt{\frac{2\ln v}{n_v}} \tag{6}$$

where $H(n)$ is the average value stored in history table and $\alpha = n_v/k$.

3.5.2 Rollout Biasing

In this heuristic we use the history table to bias the random rollouts in Eqn. 4. Rather than choosing pure random actions, we preferentially choose actions which have had a higher evaluation score as stored in the history table.

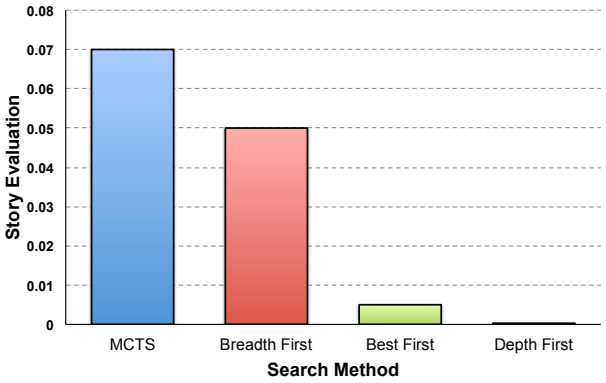
4. RESULTS

We tested our approach on an instance of the crime story domain described above. We utilized 5 actors (including 1 policeman and 1 detective), 5 places, and 5 items. The story goal is set as 2 people dead and the murderer arrested. Because each actor can use multiple items and travel to different places the resulting search space was fairly large with an average of 50 total actions available across all the actors

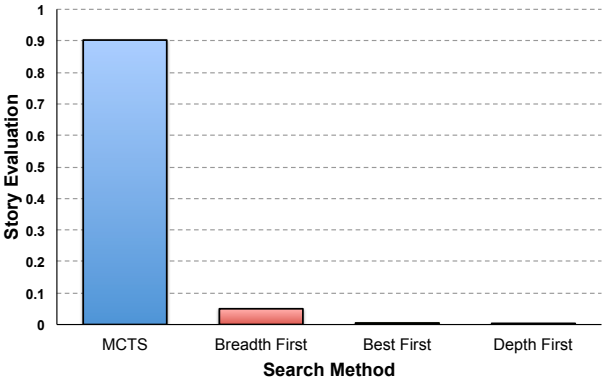
at any given time (resulting in a search tree with an average branching factor of 50).

4.1 Search Method Comparison

We first compare our method to the traditional search algorithms of Breadth-First Search, Depth-First Search, and Best-First Search. We chose these search algorithms because, like MCTS, none of them require a search heuristic. Furthermore, Breadth-First Search and Best-First Search algorithms are guaranteed to find an optimal solution given sufficient time and memory. Additionally, Best-First Search and Depth-First Search will explore longer paths earlier which can potentially find optimal solutions earlier in the search process. All search algorithms are implemented such that they maximize score from Eqn. 4. Figure 1 shows a comparison of the best story found by the different methods both for small and large search budgets (results averaged over 3 trials).



(a) Low Budget (100K Nodes)



(b) High Budget (3 Million Nodes)

Figure 1: Comparison of Search Methods Our proposed approach using Monte Carlo Tree Search (MCTS) outperforms other search techniques such as Breadth-First Search, Depth-First Search, and Best-First Search. (a) Even for a small search budget, MCTS outperforms other methods (b) The gains improve dramatically for larger budgets.

Depth-First search was observed to use very little memory, however, it failed to find narratives which met any goals. Best-First search suffers from delay caused by trying to accomplish the goals through a set of believable actions due to its high exploratory behavior. As a result, it tends to require higher budget to eventually find the optimal solution.

Alice picked up a vase from her house. Bob picked up a rifle from his house. Bob went to Alice's house. While there, greed got the better of him and Bob stole Alice's vase! This made Alice furious. Alice pilfered Bob's vase! This made Bob furious. Bob slayed Alice with a rifle! Bob fled to downtown. Bob executed Inspector Lestrade with a rifle! Charlie took a baseball bat from Bob's house. Sherlock went to Alice's house. Sherlock searched Alice's house and found a clue about the recent crime. Bob fled to Alice's house. Sherlock wrestled the rifle from Bob! This made Bob furious. Sherlock performed a citizen's arrest of Bob with his rifle and took Bob to jail.

Figure 2: High Scoring Story (Score: 0.68)

Sherlock moved to Alice's House. An Earthquake occurred at Alice's House! Sherlock and Alice both died due to the earthquake.

Figure 3: Low Scoring Story (Score: 0.016)

While Breadth-First search outperforms the Best-First search and Depth-First search methods, it is unable to find a believable means to achieve the goal even with a budget of several million nodes. In contrast, our MCTS approach outperforms all the other search techniques for both small and large budgets, and is able to find a high score story.

The difference in narratives generated by the various search approaches is highlighted in the illustrative sample narratives in Figures 2 and 3. These narratives are direct outputs from our code. We note that we automatically combine two consecutive related actions into a single sentence to improve readability of the narratives.

Figure 2 shows a sample of a high quality story, that has been generated by our MCTS algorithm. The story achieves the goals while containing several plausible actions (such as revenge killing).

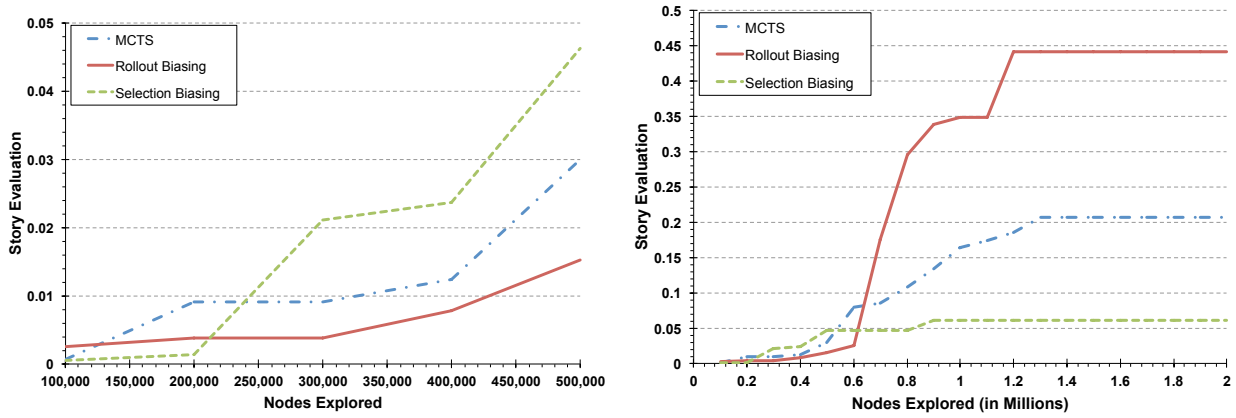
Figure 3 shows a story found by Breadth-First search. While the story is short and accomplishes the goal of two people being killed, it fails to achieve the more complex goal of somebody being arrested. Furthermore, the story makes use of an earthquake to reach its goals, which has a very low believability score.

4.2 Heuristic Comparison

We also experimented to determine the effect of our two proposed heuristics on search performance. Figure 4 summarizes our results (averaged over 3 trials). For low search budgets, the selection biasing heuristic improves performance over standard MCTS (Fig 4(a)). However, this heuristic gets stuck at a local minima and fails to improve the story even with large search budgets. In contrast, the rollout biasing heuristic leads to a substantial improvement over standard MCTS for large search budgets (Fig 4(b)).

4.3 Large Scale Scenarios

While the MCTS approach we described in Algorithm 1 works well, it consumes large amounts of memory. This large



(a) Low Budget (b) High Budget

Figure 4: Effect of Heuristics (a) For small search budgets (<500K nodes explored) the search heuristics tested had only a moderate effect on performance. (b) For large search budgets, the advantage of the rollout biasing heuristic can be clearly seen. Additionally, while the selection bias heuristic helps with small budgets it tends to get stuck in local minima.

memory usage can restrict its applicability on very large scenes. To illustrate this limitation, we extend the crime story domain above to contain 20 actors, 7 places, and 7 items. This increases the branching factor to 150 potential actions on average.

Figure 5 compares standard MCTS with our iterative approach described in Algorithm 2. Importantly, the non-iterative approach fails to complete its execution when the search budget is larger than 5 million nodes. This failure happens because the non-iterative approach is using over 100GB of memory for such large search trees. In contrast, our proposed iterative approach produce better results for lower budgets, and can run much larger budgets without failure. In fact, we were able to run with a budget over 50 million nodes on the same machine with no memory issues.

Runtime For the 5 actor story domain, our method was able to find detailed stories in under 5 seconds, and find the optimal story in less than 1 minute (using a single core on an Intel 2.2 GHz laptop processor). For the 20 actors story domain, stories took much longer to generate, though a high quality story could generally be found in under 1 hour with the iterative approach.

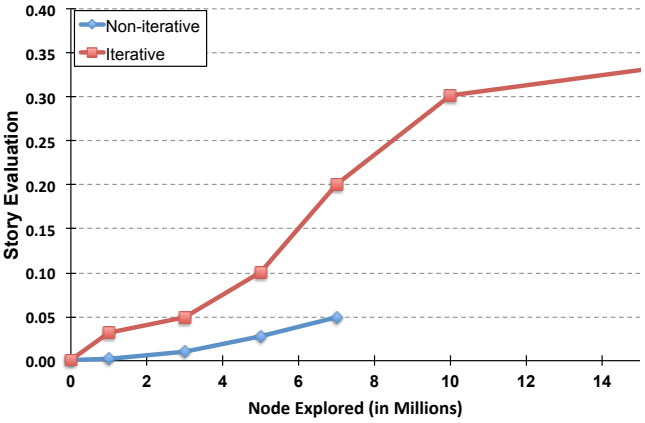


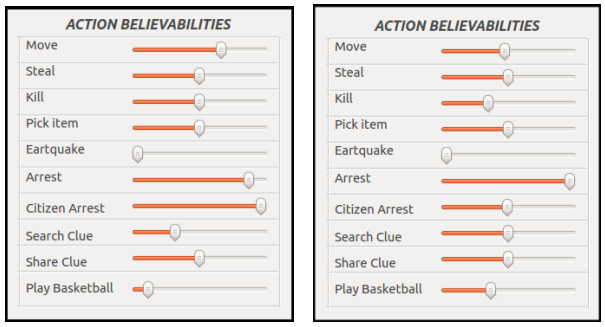
Figure 5: Iterative vs Non-iterative For very large story domains, MCTS can run out of memory trying to store the entire search tree. In the 20-person domain, the non-iterative approach could only explore trees up to 5 Million nodes before failing. Our proposed iterative approach uses tree pruning to reduce memory and can explore much large trees (producing higher value narratives).

5. USER-DRIVEN NARRATIVE VARIETY

We have developed an interactive framework that can generate narratives with parametric narrative goals and configurable context-sensitive believability metrics for each available action. Our framework employs a graphical user interface (see Appendix) which allows users to modify the believability of various actions by dragging several sliders. By changing believability of actions, users can influence the actions present in generated narratives via Eqn 1. For example, the user can generate a narrative where the goal is preferentially accomplished through earthquakes by increasing the believability of the earthquake action. In this case, given the story goal of "two people dead", the resulting story is similar to the story presented in Figure 3.

As additional examples, we present two user-selected believability configurations in Figure 6. In the choice of believabilities shown in Figure 6(a), the user set the believability of "eartquake" and "play basketball" actions to a very low value, and the believability of "Citizen arrest" is set very high. Ideally, these choices of believabilities should generate a story where the arrest is performed by a citizen. As Figure 7 shows, the resulting story meets this expectation. In contrast, the second believability setup (shown in Figure 6(b)) has a lower belivability of citizen's arresting each other. The resulting story is shown in Figure 8, with the arrest ultimately made by Inspector Lestrade. Note that the two stories have similar beginnings, but they are resolved in different fashions in accordance with the user specified believabilities.

Further options are presented to the user to modify other parameters associated with our narrative generation approach. For example, selecting between iterative and non-iterative, setting the budget of maximum nodes explored, and other similar parameters. This allows the user to control the trade-off between story quality and story generation time.



(a) Citizen arrest configuration

(b) Officer arrest configuration

Figure 6: Users can set believability of actions to generate diversity in generated stories.

Alice got a vase from her house. Bob picked up a rifle and a baseball bat from his house. Sherlock stopped by the basketball court on his way to Alice's house. Charlie went to Alice's house. Charlie took Alice's vase! This made Alice furious. Alice stole Charlie's vase! This made Charlie furious. Alice killed Charlie with a flower vase, very interesting! Bob went to Alice's house. Meanwhile Sherlock went to the basketball court. Inspector Lestrade stopped by Bob's house on his way to Alice's house. Alice fled to the basketball court. Bob searched Alice's house and found a clue about the recent crime. Sherlock went to Bob's house. Meanwhile Bob went to downtown. Alice stopped by Alice's house on her way to Bob's house. Inspector Lestrade went to downtown. Meanwhile Bob went to Alice's house. Alice executed Sherlock with a flower vase, very interesting! Bob went to Bob's house. Bob arrested Alice with his baseball bat and took Alice to jail.

Figure 7: Story generated from believability setup shown in Figure 6(a). In this configuration, the believability of "citizen arrest" action is set high, resulting Bob arresting the murderer.

6. CONCLUSION

We have presented a framework capable of generating believable narratives which satisfy user-defined goals from large story domains. By using Monte Carlo Tree Search, we were able to balance exploiting the most promising branches along with exploring other potentially good choices at each level of the tree. The resulting framework generates complex, believable narratives with only a few seconds of computation time for small domains, and a few minutes for larger ones. We also introduced a user-friendly tool that can be used by authors and teachers to generate full or partial narratives for specific scenes.

Alice secured a vase from her house. Bob secured a baseball bat and a rifle from his house. Bob went to Alice's house. Alice pilfered Bob's rifle! This made Bob furious. Inspector Lestrade went to the basketball court. Bob slayed Alice with a baseball bat! Bob slayed Charlie with a baseball bat! Bob fled to the basketball court. Meanwhile Sherlock went to Bob's house. Sherlock went to Alice's house. Meanwhile Inspector Lestrade went to Bob's house. Inspector Lestrade went to Alice's house. Inspector Lestrade searched Alice's house and found a clue about the recent crime. Inspector Lestrade went to the basketball court. Meanwhile Bob went to Alice's house. Bob stopped by the basketball court on his way to downtown. Bob and Sherlock both went to the basketball court. Sherlock picked up a basketball from the basketball court. Inspector Lestrade arrested Bob with his police gun and took Bob to jail.

Figure 8: Story generated from believability setup shown in Figure 6(b). In this configuration, the Inspector Lestrade arrests the murder due to low believability of "citizen arrest" action.

Limitations: While our method is capable of exploring large story domains, our approach still has some limitations. The size of the tree being stored in memory still grows exponentially as the number of potential actions increases. Therefore, a narrative involving hundreds of characters is likely to run out of memory on consumer hardware. We have also focused only on domain-independent heuristics. Usage of domain specific heuristics can alleviate memory problems and reduce runtime. Another limitation of our work is that defining believability for actions must be done by the user and can become onerous as the number of actions increases. We aim to resolve this issue by utilizing a data-driven approach in order to discover believability of actions automatically.

Future Work: Beyond addressing the above limitations, we think there are exciting directions for future work. We would like to explore other forms of evaluation criteria beyond our believability metric. For example, a user might want to specify the pacing of a narrative to ensure rising action and climax which might require dynamically changing believability parameters during the narrative generation, or use of a different metric altogether.

Acknowledgments: This work was supported in part by the University of Minnesota Supercomputing Institute.

7. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning,Springer*, 47(2-3):235–256, 2002.

[2] H. Barber and D. Kudenko. Generation of dilemma-based interactive narratives with a changeable story goal. In *INTETAIN*, 2008.

[3] M. Brenner. Creating dynamic story plots with continual multiagent planning. In *AAAI*, 2010.

[4] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener,

D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Trans. on Computational Intelligence and AI in Games, IEEE*, 4(1):1–43, 2012.

[5] M. Cavazza, F. Charles, and S. J. Mead. Character-based interactive storytelling. 2002.

[6] T. Cazenave. Reflexive monte-carlo search. In *Computer Games Workshop*, pages 165–173, 2007.

[7] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-carlo tree search: A new framework for game ai. In *AIIDE*, 2008.

[8] G. Chaslot, S. De Jong, J.-T. Saito, and J. Uiterwijk. Monte-carlo tree search in production management problems. In *BNAIC*, pages 91–98, 2006.

[9] M. Enzenberger, M. Muller, B. Arneson, and R. Segal. Fuego—an open-source framework for board games and go engine based on monte carlo tree search. *IEEE Trans. on Computational Intelligence and AI in Games, IEEE*, 2:259–270, 2010.

[10] M. Fox and D. Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[11] P. Gervás, B. Díaz-Agudo, F. Peinado, and R. Hervás. Story plot generation based on cbr. *Knowledge-Based Systems, Elsevier*, 18(4):235–242, 2005.

[12] M. Helmert, G. Röger, J. Seipp, E. Karpas, J. Hoffmann, E. Keyder, R. Nissim, S. Richter, and M. Westphal. Fast downward stone soup. *Seventh International Planning Competition*, pages 38–45, 2011.

[13] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

[14] N. McIntyre and M. Lapata. Plot induction and evolutionary search for story generation. In *ACL*, pages 1562–1572. ACL, 2010.

[15] S. Ontanon and J. Zhu. The sam algorithm for analogy-based story generation. In *AIIDE*, 2011.

[16] R. Pérez Ý Pérez and M. Sharples. Mexica: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence*, 13(2):119–139, 2001.

[17] J. Permar and B. Magerko. A conceptual blending approach to the generation of cognitive scripts for interactive narrative. In *AIIDE*, 2013.

[18] S. Richter and M. Westphal. The lama planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39(1):127–177, 2010.

[19] M. O. Riedl and R. M. Young. An intent-driven planner for multi-agent story generation. In *AAMAS*, pages 186–193, 2004.

[20] M. O. Riedl and R. M. Young. An objective character believability evaluation procedure for multi-agent story generation systems. In *IVA*, pages 278–291. Springer, 2005.

[21] M. O. Riedl and R. M. Young. Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research*, 39(1):217–268, 2010.

[22] S. Samothrakis, D. Robles, and S. M. Lucas. A uct agent for tron: Initial investigations. In *CIG*, pages 365–371, 2010.

[23] A. Shoulson, M. Gilbert, M. Kapadia, and N. I. Badler. An event-centric planning approach for dynamic real-time narrative. In *MIG*, 2013.

[24] D. Silver and J. Veness. Monte-carlo planning in large pomdps. In *NIPS*, pages 2164–2172, 2010.

[25] J. Skorupski, L. Jayapalan, S. Marquez, and M. Mateas. Wide ruled: A friendly interface to author-goal based story generation. In *Virtual Storytelling. Using Virtual Reality Technologies for Storytelling*, pages 26–37. Springer, 2007.

[26] B. Srivastava, T. A. Nguyen, A. Gerevini, S. Kambhampati, M. B. Do, and I. Serina. Domain independent approaches for finding diverse plans. In *IJCAI*, pages 2016–2022, 2007.

[27] I. M. T. Swartjes and M. Theune. The virtual storyteller: story generation by simulation. In *BNAIC*, pages 257–265, 2008.

[28] J. Teutenberg and J. Porteous. Efficient intent-based narrative generation using multiple planning agents. In *AAMAS*, pages 603–610, 2013.

[29] M. Theune, S. Faas, E. Faas, A. Nijholt, and D. Heylen. The virtual storyteller: Story creation by intelligent agents. In *TIDSE*, pages 204–215, 2003.

APPENDIX

A. GRAPHICAL USER INTERFACE

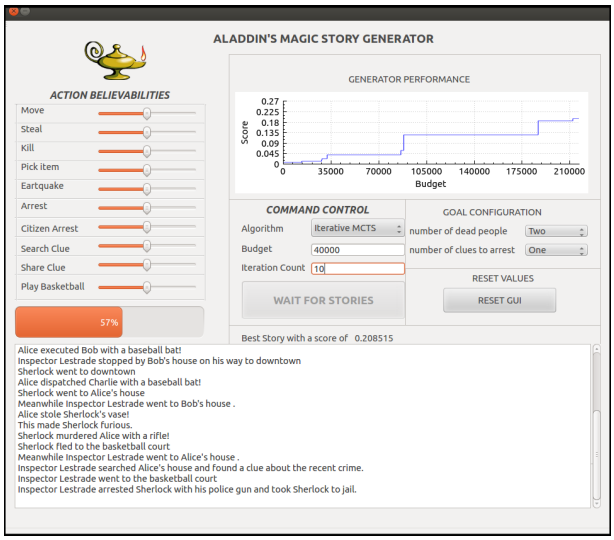


Figure 9: Graphical User Interface (GUI)

Figure 9 shows a graphical user interface utilizing our story generation framework. Users can modify believabilities of various actions, set story goals, select a planning strategy, and choose a planning budget. As the generation process unfolds, the best story found so far is displayed along with a graph of the story evaluation score progress.

A video showing our system running can be found at: `http://motion.cs.umn.edu/r/StoryMCTS`