Language Models as Narrative Planning Heuristics

Lasantha Senanayake
Narrative Intelligence Lab
University of Kentucky
Lexington, Kentucky, USA
lmse234@uky.edu

Stephen G. Ware
Narrative Intelligence Lab
University of Kentucky
Lexington, Kentucky, USA
sgware@cs.uky.edu

Abstract

Narrative planning is the process of generating sequences of actions that form coherent and goal-oriented narratives. Classical implementations of narrative planning rely on heuristic search techniques to offer structured story generation but face challenges with scalability due to large branching factors and deep search requirements. Large Language Models (LLMs), with their extensive training on diverse linguistic datasets, excel in understanding and generating coherent narratives. However, their planning ability lacks the precision and structure needed for effective narrative planning. This paper explores a hybrid approach that uses LLMs as heuristic guides within classical search frameworks for narrative planning. We compare various prompt designs to generate LLM heuristic predictions and evaluate their performance against h^{+}, h^{max}, and relaxed plan heuristics. Additionally, we analyze the ability of relaxed plans to predict the next action correctly, comparing it to the LLMs' ability to make the same prediction. Our findings indicate that LLMs rarely exceed the accuracy of classical planning heuristics.

CCS Concepts

• **Computing methodologies → Search methodologies**; **Multi-agent planning**; *Machine learning*.

Keywords

Narrative Planning, Narrative, Planning, Heuristic Search, Large Language Models, Neuro-Symbolic

ACM Reference Format:
Lasantha Senanayake and Stephen G. Ware. 2025. Language Models as Narrative Planning Heuristics. In *International Conference on the Foundations of Digital Games (FDG '25), April 15–18, 2025, Graz, Austria.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3723498.3723815

Introduction

Interactive narrative environments, such as games and training simulations, often need to maintain a coherent storyline while adapting to user choices. Narrative planning is one tool to generate and regenerate structured stories during runtime [31].

Sabre [27] is a narrative planner developed to address the complexities of multi-agent storytelling in virtual environments. It is a narrative planning system that models both the intentions of the author and the intentions and beliefs of each virtual character. This allows Sabre to generate plans that improve the author's utility while ensuring that every action taken by a character can be explained by that character's own goals and beliefs. This makes it particularly suitable for generating interactive narratives where character realism and coherent story progression are crucial.

Sabre, and other systems based on classical planning, provides an approach to story generation that reasons about the preconditions and effects of events using formal reasoning similar to predicate logic [30]. Unlike reactive systems that select individual actions with minimal foresight, planning-based systems explore many possible story paths to ensure both quality and structure. However, planning is P-SPACE hard [9] and does not scale to long stories. Heuristic search can improve scalability. For example, the Glaive heuristic [28] has been specifically designed for narrative search. However, deep searches will always become intractable eventually.

LLMs, such as GPT-4o mini [3], are neural transformers trained on various natural language datasets. They can capture story structure, character development patterns, and genre expectations in their latent space. While LLMs are proficient at understanding and generating natural language, they are not inherently structured in their approach to maintaining narrative coherence over long sequences of actions.

This paper lays the groundwork for leveraging the strengths of both LLMs and classical heuristic search methods in narrative planning. Our long-term goal is to use LLMs as heuristics within a search framework, combining the narrative understanding of LLMs with the formal structure of classical planning methods. This hybrid approach has the potential to generate coherent and compelling stories more efficiently. This paper presents a study that investigates whether LLMs can provide accurate estimates of story size and content, serving as an initial step toward our objective.

Our findings indicate that LLMs can sometimes achieve accuracy comparable to classical planning heuristics, but the results were mixed. For some settings in some scenarios, LLMs matched or slightly exceeded the ability of heuristics to estimate the number of actions remaining in a story and which next actions can lead to solutions. In others, their predictions deviated significantly. These results suggest that, while LLMs have potential as heuristic tools in narrative planning, their performance is not yet reliable enough to replace existing heuristics.

Related Work

Planning algorithms have a long history in the games industry, from Orkin's Goal-Oriented Action Planning (GOAP) framework [15] to multi-agent tactics planning [5], to automated game testing [20].

This work is licensed under a Creative Commons Attribution International 4.0 License.

FDG '25, Graz, Austria
© 2025 Copyright held by the owner/author(s).
ACM ISBN /25/04
https://doi.org/10.1145/3723498.3723815

Planning is popular for generating interactive stories in games because it provides a formal, generative framework that reasons about causality and event ordering [30]. Planning has been used to make playable interactive versions of *Friends* [4], *Madame Bovary* [17], and *The Merchant of Venice* [19]. It has been used in interactive training simulations [8]. Young et. al provide a survey of other story planning systems [31].

In interactive narrative research, ensuring structured and coherent story generation has led to the development of various storytelling algorithms. Early systems such as Tale-Spin [14], Universe [12], and Façade [13], employed symbolic models using predicate logic to represent characters, locations, objects, and conditions. These systems shared a common approach of using preconditions to limit when actions could occur and effects to describe how actions change the narrative state, but they did not perform extensive searches for plans that meet structural requirements. Tale-Spin and Universe are not true planning systems, and Façade has limited planning capabilities.

Advancements in narrative planning have seen a focus on integrating characters' goals and intentions. Systems such as *IPOCL* [21] emphasize character intentions to drive the plot while maintaining coherence. *Glaive* [28] and *IMPACTical* [24] extend that model to allow for conflict and failed plans. *HeadSpace* [23], *Ostari* [6], and *Sabre* [27] generate stories where characters can operate under mistaken beliefs, leading them to attempt actions that fail.

Recent efforts have explored the integration of LLMs and narrative planning. In *Neural Story Planning* [29] Ye et al. use a technique similar to partial-order planning to keep an LLM goal-oriented during story generation. In *Planning Stories Neurally* [7] Farrell and Ware used an LLM as a cost function during planning, giving next actions suggested by the LLM lower weight to ensure they are visited sooner in the search. Our method is similar to theirs because we are using an LLM to guide a narrative planner, but whereas they only used the LLM to suggest the immediate next action, we ask the LLM to fully complete the story so that it can serve as a heuristic that measures the remaining distance to the goal.

On the Planning Abilities of Large Language Models [25] evaluates the use of LLMs as heuristic guides in the Blocksworld planning domain, demonstrating their potential to drive sound plans. Although not specifically focused on narrative problems, the techniques employed are transferable to narrative contexts, showcasing the versatility of LLMs in planning tasks.

Background

In forward state-space planning [22], actions are sequentially added to the end of a plan until a goal state is achieved. A plan that successfully reaches a goal state is a solution. However, narrative planners have extra constraints on solutions to ensure that they meet desired narrative criteria, such as characters acting believably. To address this, narrative planners can define characters as specific entities with their own goals, who can only engage in actions that help achieve these goals.

We use the Sabre narrative planner [27], which incorporates character beliefs and intentions so that the plans characters form align with their beliefs and make them appear to have limited observability. In the case of Sabre, a plan is considered a solution if it achieves the author's goal and includes only actions that are explained.

The detailed definition of action explainability in Sabre is provided by Ware and Siler [27], but a brief summary is as follows: an action is explained if it is explained for each consenting character involved. For a consenting character, an action is explained if it is the first step in a plan which can be executed in the state that character believes they are in to improve that character's utility.

During the search process, the planner monitors when character goals are met and propagates explanations backward, marking previous actions as explained if they contributed to achieving the goals. To plan efficiently in large spaces, it is crucial for a planner to prioritize actions that are likely to lead to a solution.

For some given state, a classical planning heuristic estimates the number of actions that need to be taken to reach a goal state. In narrative planning, simply reaching the goal state is insufficient; actions must also be explained for the characters who take them. The intuition behind this paper is that asking an LLM to complete a story will serve as a better heuristic than those used by classical planners because the actions suggested by the LLM are more likely to be explained actions. In other words, an LLM's suggestions are more likely to account for each character's beliefs and motivations.

We compare LLM suggestions to three popular planning heuristics on a suite of benchmark narrative planning tasks. All three heuristics share a common approach of solving a relaxed version of the planning problem and using the size of a relaxed solution as an approximation for the distance to the goal. The relaxed problem assumes that once a proposition is true is can never become false again. This is sometimes referred to as "ignoring delete lists."

The h^+ and $h^{\max}$ heuristics [2] define the cost of any proposition that is true in the given state as 0. The cost of all other propositions is 1 plus the cost of the lowest cost precondition of any action which has that proposition as an effect. The h^+ heuristic defines the cost of a conjunction as the sum of the costs of its conjuncts. The $h^{\max}$ heuristic defines the cost of a conjunction as the maximum cost of its conjuncts.

The h^{rp} heuristic is based on Fast Forward [10]. It behaves similarly to $h^{\max}$, but while calculating its cost it uses a plan graph [1] to build a solution to the relaxed planning problem. Ideally, that relaxed plan will be similar to the actual solution in both length and content.

Problem Definition

A Sabre narrative planning problem has several components:

- **Characters**: Special entities that exhibit beliefs and intentions.
- **State Fluents**: Properties that change over time, which can be Boolean, nominal, or numeric. Sabre allows for a range of logical operators on these fluents and supports the *believes* modality so the planner can reason about what characters believe, what they believe others believe, etc.
- **Actions**: Modeled similarly to ADL operators [16], actions specify preconditions for their execution and effects that alter the world state. Each action includes a set of consenting characters who willingly participate in the action and

an observation function to determine which characters are aware of the action when it occurs.

- **Triggers**: These operate like actions but occur automatically. Once their preconditions are met within the state, their effects are applied immediately without requiring consenting characters or observation functions.
- **Utility Functions**: These define preferences over different states. This includes the author utility function, which represents the desired states the planner aims to achieve (the story goal), and individual utility functions for each character to represent their personal goals.
- **Initial State**: Gives initial values for all state fluents and details any incorrect beliefs that characters initially hold.

The objective is to transition from the initial state to a state where the author utility is increased by only taking actions which are explained for all of the consenting characters who take them. For some given state, a heuristic estimates the number of actions that need to be taken before the goal is reached.

Method

State Distance Data Collection

To evaluate whether a language model can be used as a narrative planning heuristic, we first need a dataset of states from narrative planning where the distance to a solution is known. To collect this set of states, we conducted breadth-first searches on a suite of benchmark narrative planning problems collected from various sources [26].

Table 1 shows the size of each benchmark problem, represented by the number of characters $|C|$, fluents $|F|$, actions $|A|$, and triggers $|T|$ after grounding. It also lists three relevant parameters for the breadth-first search of that problem. **ATL** is the Author Temporal Limit and defines how many actions may appear in a solution to the problem. The ATL for each problem was determined experimentally. Each search was allowed to run up to three days on a computer with an Intel Xeon 4.1 GHz processor and 512GB RAM. The ATL for each search was limited to the depth it could fully explore in that time. **CTL** is the Character Temporal Limit and defines the number of actions that can appear in a character's plan when they explain why they took an action. The CTL for each search was chosen based on known values for each problem explained in the benchmark suite [26]. **EL** is the Epistemic Limit and defines how deeply the characters reason about theory of mind. An EL of 1 means the planner reasons only about what characters believe; an EL of 2 means characters reason about what they believe and what they believe others believe, and so on. EL was also chosen based on known values from the benchmark suite.

It is possible to run deeper searches and to generate more solutions by using heuristic search, however, we feared this would bias the set of states collected toward states where classical heuristic estimates are more accurate. Breadth-first search also has the benefit of ensuring we find shortest solutions.

Three benchmark problems were not able to reach a depth that contained any solutions, so those have been excluded from our experiments.

Table 1: Details for Narrative Planning Benchmark Problems

Problem	Problem Size				Configuration		
	$\|C\|$	$\|F\|$	$\|A\|$	$\|T\|$	**ATL**	**CTL**	**EL**
Bribery	3	16	27	0	5	5	2
Deer Hunter	3	35	28	76	10	6	1
Fantasy	4	68	76	136	8	3	2
Gramma	4	61	812	896	7	5	2
Hospital	4	57	102	196	4	5	3
Jailbreak	3	26	106	54	7	6	1
Lovers	3	40	312	375	7	5	2
Raiders	3	21	39	66	8	4	1
Secret Agent	2	14	44	75	9	8	1
Space	2	23	29	62	9	3	1
Treasure	2	4	5	0	4	4	3

After each search, we recorded every state which has a known distance to a solution that could be found under the above constraints. We recorded both author and character states. For example, the shortest solution to improve the author's utility in the *Gramma* problem is: Tom walks to the crossroads. The bandit walks to the crossroads. The bandit kills Tom. The initial state of the problem and the state after each of these actions was recorded. The state after the first action has a distance to the goal of 2. The state after the second action has a distance to the goal of 1, and so on. We also recorded the Bandit's plan to improve her utility: The bandit walks to the crossroads. The bandit kills Tom. The bandit loots Tom's coin. We recorded each of these states and their distance to the Bandit's goal, and so on for Tom and all other characters.

Comparing LLM Predictions with Classical Heuristics

For each of the states collected above, we describe the story so far and the goal to an LLM and ask it to complete the story. We compare the plans returned by the LLM to the classical heuristics h^+, $h^{\max}$, and h^{rp}. For all states, we compare:

- The ground truth minimum number of explained actions that need to be taken to reach a state where utility is higher for the author (or where utility is higher for the character who is planning, if this is a character belief state), as discovered by the breadth-first search above
- The number of actions that each heuristic estimates need to be taken to reach a state where utility is higher
- The number of actions in the plan returned by the LLM after it was asked to complete the story

Recall that h^{rp} generates a relaxed plan. Heuristics that generate a plan are especially useful, since heuristic searches often face many ties. Planners like Fast Forward bias their searches toward the actions that appear in relaxed plans.

When evaluating h^{rp}, we can compare both the length of its relaxed plans and the content of those plans to the plan generated by the LLM. Using the states recorded above, we can determine which actions are known to lead to solutions. To continue the previous example, after Tom walks to the crossroads, the action where the bandit walks to the crossroads is known to lead to a solution. There may be multiple such actions.

If the plan generated by a heuristic contains an action that is known to lead to a solution then it is an especially good plan because it can be used to bias the search toward solutions. When comparing h^{rp} to the LLM, we also measure how frequently the plans they returned contained at least one action known to lead to a solution. Note this is a conservative metric because other actions might lead to solutions that were out of reach for breadth-first search.

Prompt Design

To use an LLM (specifically GPT-4o mini [3]) as a narrative planning heuristic, we created a prompt template that provides context about the story so far and asks for the next actions to achieve the author's goal or the goal of the character who is planning. An example prompt is given in Figure 1. The text colors in that figure correspond to each part of the template:

- **Task**: We tell the LLM its purpose is to complete a story. This text was the same for every prompt across every problem.
- **Problem**: We list all of the objects that exist in the problem and the properties they can have. We introduce each character and explain their goals. We describe each kind of action that can occur, explaining any preconditions or effects that might be non-obvious. This text was written by hand for each problem and was the same for every prompt for that problem.
- **Previous Actions**: We list any actions that have occurred so far, translating each into a natural language sentence. The story so far is described from the perspective of the character who is planning. Since Hawkins is the character who is planning in this example, we list only actions that Hawkins has observed. This text was different and generated automatically for each prompt.
- **Current State**: We fully describe the current state, giving the value for each fluent (including beliefs) as a natural language sentence. The current state is described from the perspective of the character who is planning; e.g. we describe the state that Hawkins believes to be the case. This text was different and generated automatically for each prompt.
- **Instructions**: We instruct the LLM to complete the story using as few actions as possible and without adding new elements or actions to the problem. This text was the same for every prompt across every problem.
- **Goal**: We tell the LLM which of the character's goals to complete. Because Hawkins is the character who is planning, we tell the LLM to achieve Hawkins' goal. If we are planning for the author's goal, the author's goal is described instead, e.g. "Give me the shortest story that ends with Hawkins having the treasure." This text was different and generated automatically for each prompt.
- **Formating Instructions**: We tell the LLM to format its answer as a JSON object for easy parsing of the result. This text was the same for every prompt across all problems.

We used Zero-Shot Chain-of-Thought (CoT) prompting [11] to construct this template prompt. Zero-Shot CoT prompting allows the LLM to break down its reasoning step by step without prior examples, enhancing the clarity and structure of the response.

Problem Translation. Text describing the objects, fluents, characters, and actions in each problem are, in most cases, directly based on the problem descriptions [26]. However, for the *Hospital* and *Lovers* problems we translated some symbols into specific objects. For example, the agents in the original *Lovers* problem are named $C1$, $C2$, etc. A planner can easily treat these as unique symbols, but to make this problem more suitable for an LLM, we named the agents Alex, Blake, etc. Similarly, we renamed the items from $I1$, $I2$, etc. to flowers, chocolate, etc. In the *Hospital* problem, we gave the diseases specific fictional names and described specific symptoms.

Known Problems. At least two problems, *Treasure* and *Raiders*, are based on the well-known stories *Treasure Island* and *Raiders of the Lost Ark* respectively. Summaries of these stories likely appear in the LLM's training data. This may have influenced the LLM's performance on these problems.

Prompt Variations

We experimented with several variations on our prompt template. Four were of particular interest and are evaluated in this paper.

Prompt with Natural Language. This variant converts the objects, characters, fluents, and actions from Sabre into natural language. For instance:

- `location(Gargax) = Cave` becomes *Gargax is at the Cave.*
- `believes(Talia, alive(Gargax)) = True` becomes *Talia believes Gargax is alive.*
- `travel(Talia, Village, Cave)` becomes *Talia travels from the Village to the Cave.*

This method, which is reflected in Figure 1, is referred to as *LLM Natural*. It ensures that the input and expected output are in a format easily understood by the LLM.

Prompt with Syntax. In the second method, we constructed the prompt using the original Sabre syntax, maintaining the initial state, action, and plan format. Instead of translating `location(Gargax) = Cave` into natural language, we simply expressed propositions in that format. The special text explaining the problem in natural language (the red text in Figure 1) was still used at the start of the prompt. To our surprise, the LLM was often able to interpret Sabre syntax correctly, and this prompt design requires less special-purpose code for translation. This method, referred to as *LLM Syntax*, tests whether LLM can understand and provide predictions directly in Sabre language.

Handling Lengthy Outputs. To address the tendency of LLMs to produce verbose responses, we devised two additional prompt types to help limit the length of the response. We implemented a method to calculate the remaining number of steps needed to reach a solution by subtracting the number of actions that have occurred already from a predefined maximum plan length. This predefined maximum is based on the depth reached during that problem's breadth-first search.

I will describe a setting and the first part of a story. Your job is to complete the story to ensure it has a specific ending.

There are two locations in this story: the port and the island. There is one item in this story: some treasure. There are two characters in this story. Hawkins is a boy who wants the treasure. Silver is a pirate who wants the treasure. There are four kinds of actions characters can take in the story. If the treasure is buried on the island, Hawkins can spread a rumor that will make Silver believe the treasure is buried on the island. Hawkins and Silver can work together to sail a ship from the port to the island. If the treasure is buried on the island, Hawkins can dig up the treasure. A character can take the treasure once it has been dug up.

These events have already happened in the story: Hawkins spreads a rumor that the treasure is buried on the island. Hawkins and Silver sail from Port Royal to the island. Hawkins digs up the treasure.

This is the current situation after those events: Hawkins is at the island. Silver is at the island. The treasure is at the island. Silver believes that the treasure is at the island.

Complete the story using only these locations, items, characters, and actions. Do not invent new locations, items, characters, or actions. Characters should only take actions that help to achieve their goals, and the story should only include actions which are necessary to achieve the ending.

Give me the shortest story where Hawkins achieves their goal.

Explain why each action is in the story. After the explanation of the whole story, give a JSON object with the final plan. The JSON should include an array called 'plan' with the sequence of actions (as a string) taken to achieve the goal. Example format: {plan: ['action1', 'action2', 'action3']}.

Figure 1: An example natural language prompt for the *Treasure* problem, with key sections colored.

- **LLM Natural with Limits**: This prompt type uses natural language but specifies the maximum number of actions expected in the response.
- **LLM Syntax with Limits**: This prompt type uses Sabre syntax but but specifies the maximum number of actions expected in the response.

For both limited prompt types, the final prompt question was modified to specify the preferred plan length. An example of the prompt question with limits is as follows:

"While keeping the plan complete, a smaller plan is preferred. Suggested maximum length of the plan: {SUGGESTED_PLAN_LENGTH}."

State Sampling

In some problems, the exhaustive breadth-first search produced 10,000+ or 100,000+ states. To limit the cost of querying GPT-4o mini, we sampled 1000 states uniformly at random for each problem (or all states if a problem generated fewer than 1000). We sampled only states that had a verified non-zero distance to a goal.

Most LLMs have a temperature setting that controls the level of randomness used when generating their output. To ensure replicability, we used a temperature of 0 for all queries. In addition, we set a token limit of 3000 to control the length of responses generated by the LLM.

Parsing Responses

The LLM gives its response to each prompt formatted as a JSON object for easy parsing. However, actions were expressed as natural language sentences (for the *Natural* prompts) or occasionally as incorrectly formatted Sabre actions (for the *Syntax* prompts). In both cases, we need to translate the LLM's plan back into a Sabre plan. We use the same method as Farrell and Ware [7]. As a pre-processing step, we embed every possible action from the problem using *text-embedding-ada-002* model. When we receive a plan from the LLM, we embed each action in the same way and translate each action it into the action from the problem whose embedding has the lowest cosine distance.

Comparison to Heuristic Predictions. For each state, we obtained four different suggested plans to reach the goal using our four prompts: *LLM Natural*, *LLM Syntax*, *LLM Natural with Limits*, and *LLM Syntax with Limits*. The length of these plans were compared against the length estimates of three classical heuristics h^{+}, $h^{\max}$, and h^{rp} when evaluated on the same states for the same goals. In addition to evaluating the length of the predictions, we also measured whether the relaxed plans generated by h^{rp} and the plans generated by the LLM contained actions that were known to lead to a solution.

Heuristic Weighting. Some heuristic search strategies, like Weighted A* [18], multiply the value returned by the heuristic by some constant factor ϵ to improve performance. For example, if a heuristic consistently underestimates, multiplying it by $\epsilon > 1$ may improve its overall accuracy. When presenting the accuracy of each heuristic, we will consider every ϵ weight between 0.1 and 2 (inclusive) in increments of 0.1 so as to capture ideal versions of each heuristic.

Results

Heuristic Accuracy

First, we compare the accuracy of classical planning heuristics to our LLM heuristics, considering many possible values of the weight ϵ. The error of a heuristic on a given state is the absolute value of the difference between the known number of explained actions that needs to be taken until the goal can be achieved and the value estimated by the heuristic. We analyze the Mean Squared

Table 2: Minimum Mean Squared Error Values Fine-Tuning ϵ for Each Problem

Problem	h^{max}	h^{+}	h^{rp}	*hSyntax*	*hSyntaxLimit*	*hNatural*	*hNaturalLimit*
Bribery	**0.0000**	**0.0000**	**0.0000**	0.8789	0.4537	0.7486	0.7066
Deer Hunter	0.0511	**0.2685**	0.5022	0.6235	0.6020	0.6021	0.6868
Fantasy	**0.0070**	0.4770	0.2648	0.4922	0.5147	0.3768	0.5605
Gramma	0.1396	**0.0750**	0.4787	0.6527	0.5070	0.6229	0.5360
Hospital	0.0926	**0.0380**	0.5365	0.6539	0.5638	0.6298	0.5941
Jailbreak	0.3118	**0.1520**	0.2935	1.7975	1.7661	1.8149	1.9105
Lovers	**0.1127**	0.1333	0.3651	0.6428	0.7306	0.4924	0.6858
Raiders	0.7329	**0.1648**	0.4981	1.9185	1.5562	1.6951	0.9366
Secret Agent	1.3323	1.4759	2.5128	5.2686	**0.7517**	3.6071	0.7959
Space	0.0968	**0.0661**	0.4546	0.3425	0.4160	0.3706	0.4302
Treasure	**0.0685**	0.0846	0.2692	0.2000	0.4042	1.4905	0.2778

Error (MSE) values for each each heuristic on every state for each benchmark problem.

Figure 2 illustrates how the MSE changes across different problems and ϵ weights. The x-axis shows the epsilon values, and the y-axis represents the Mean Squared Error (MSE) between the predicted and actual distances to the goal. A lower MSE means that the heuristic is more accurate. The graphs highlight that the best epsilon values can vary from one problem to another.

As seen in Figure 2, LLM-based heuristic predictions without response limits tend to give overly long plans, which results in higher MSE values. This behavior highlights the verbosity of LLMs, necessitating the use of smaller ϵ values to reduce MSE. In contrast, LLM heuristics with response limits remain more consistent, showing MSE values that are more comparable to classical heuristics, though they only surpass them on one problem.

Table 2 gives the best MSE values for each heuristic on each problem—i.e. the lowest error that can be achieved when using the best possible value of ϵ. Here are some key observations:

- **Classical Heuristics**: The classical planning heuristics generally have lower MSE values compared to the heuristics derived from LLMs, though different heuristics performed better on different problems. h^{+} shows competitive performance with the lowest MSE in several problems.
- **LLM-Based Heuristics**: The performance of the LLM-based heuristics varies significantly across problems. In many cases, they do not achieve the same level of accuracy as the simpler classical planning heuristics. For example, *LLM Syntax* and *LLM Syntax with Limits* often have higher MSE values, particularly in problems like *Secret Agent* and *Raiders*. This inconsistency suggests that the LLM-based heuristics may require further fine-tuning for specific contexts.

Heuristic Predictions of Plan Content

Next, we compare how well h^{rp} and the LLM heuristics do at predicting plans that contain actions that are known to lead to solutions. We performed two variations of this analysis.

First, Table 3 shows the percentage of times that the *first* action in the plan returned by a heuristic was an action that is known to lead to a solution. Second, since the order of actions in the relaxed plans returned by h^{rp} can be nondeterministic, Table 4 shows the percentage of times the plan returned by a heuristic contained *any* actions that are known to lead to a solution.

Table 3: Percent Accuracy of Heuristics in Predicting the First Correct Action

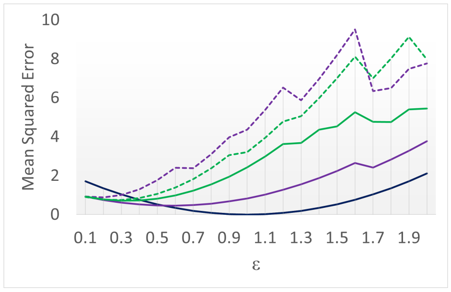
Problem	h^{rp}	*hSyn*	*hSynLim*	*hNat*	*hNatLim*
Bribery	**45.7**	17.1	37.1	14.3	14.3
Deer Hunter	**87.1**	69.3	64.3	35.7	30.3
Fantasy	**66.9**	45.4	41.7	26.8	30.3
Gramma	19.2	**24.9**	22.9	16	8.7
Hospital	**49.6**	1.5	2.1	8.7	9.2
Jailbreak	**61.5**	35.8	41.6	29.5	23.8
Lovers	**36.6**	9.1	12.7	11	13.6
Raiders	**41.5**	24.5	28.3	13.2	18.9
Secret Agent	**61.2**	34.7	49.0	26.5	30.6
Space	34.4	46.1	**52.9**	41.4	45.2
Treasure	36.8	**42.1**	36.8	36.8	**42.1**

Table 4: Frequency of Correct Actions Appearing Anywhere in Heuristic-Generated Plans

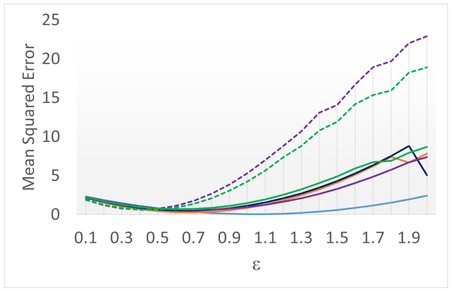
Problem	h^{rp}	*hSyn*	*hSynLim*	*hNat*	*hNatLim*
Bribery	**45.7**	25.7	37.1	14.3	14.3
Deer Hunter	**92.6**	78.3	67.9	55.1	39.1
Fantasy	**68.1**	57.5	44.8	39.3	37.1
Gramma	**30.0**	29.1	24.1	24.3	11.9
Hospital	**66.1**	3.3	5.0	16.1	14.0
Jailbreak	**69.9**	64.5	53.1	48.6	33.6
Lovers	**39.4**	14.4	13	20.3	17.2
Raiders	**49.1**	35.9	37.7	24.5	18.9
Secret Agent	**69.4**	42.9	49.0	28.6	30.6
Space	59.2	**82.6**	69.9	80	60.2
Treasure	36.8	42.1	**47.4**	42.1	42.1

Overall, according to the data we present in Tables 3 and 4, the baseline h^{rp} heuristic consistently outperforms LLM-based approaches in most problems, with notable exceptions in the *Space*, *Gramma*, and *Treasure* problems. In *Space* and *Treasure*, every LLM

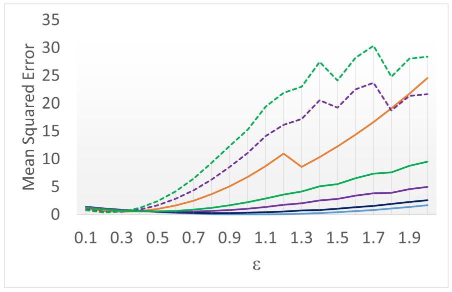
Figure 2: Impact of Epsilon Value on Heuristic Prediction Accuracy Across Problems



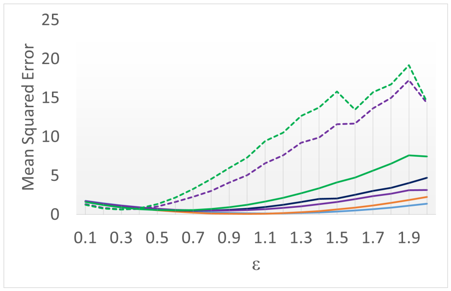
(a) Bribery



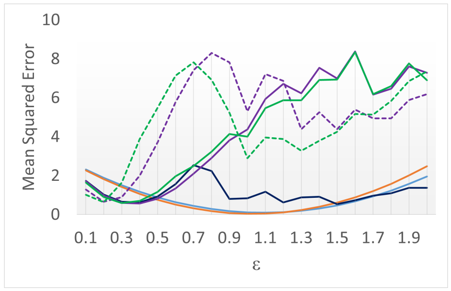
(b) Deer Hunter



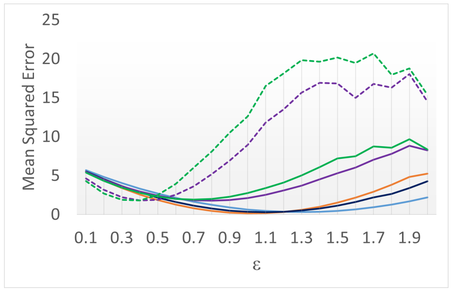
(c) Fantasy



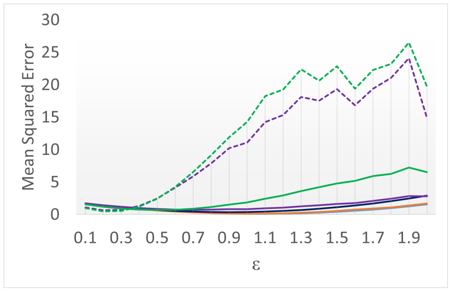
(d) Gramma



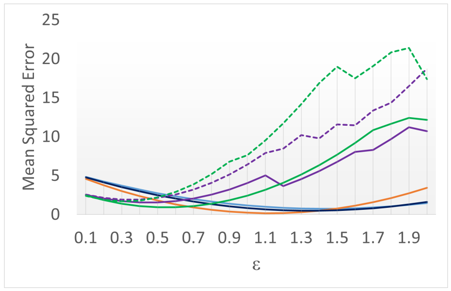
(e) Hospital



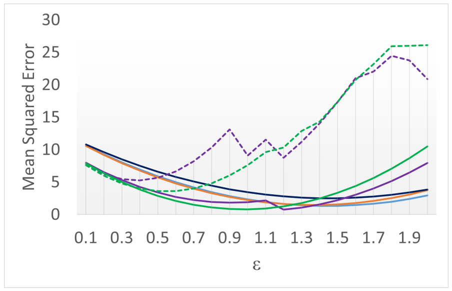
(f) Jailbreak



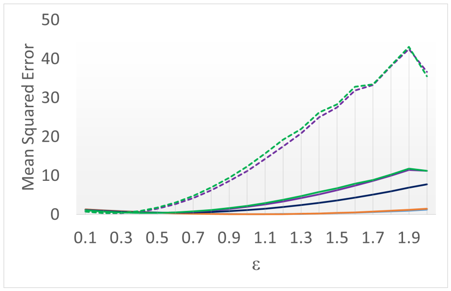
(g) Lovers



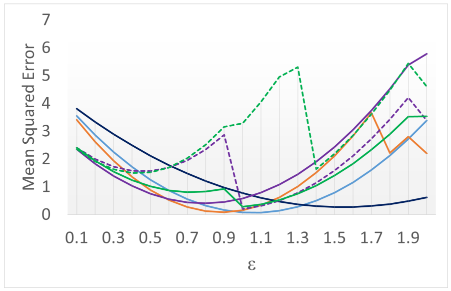
(h) Raiders



(i) Secret Agent



(j) Space



(k) Treasure

━ LLM Natural Heuristic with Limits

━ LLM Syntax Heuristic with Limits

┅ LLM Natural Heuristic

┅ LLM Syntax Heuristic

━ Relaxed Plan Heuristc

━ h_+ Heuristc

━ h_{max} Heuristc

heuristic beats or ties h^{rp} on both first action and any action prediction. In *Gramma*, the LLM Syntax heuristics beat h^{rp} on first action prediction but not any action prediction.

In the *Deer Hunter* and *Fantasy* problems, while the baseline h^{rp} retains a higher accuracy, the LLM-based predictions—especially the syntax-based ones—show promise. The *LLM Syntax* heuristic achieves 69.3% accuracy at first action prediction in *Deer Hunter*, coming closer to the baseline's 87.1%, and similarly, in the *Fantasy* problem, *LLM syntax* demonstrates a reasonable approximation to the h^{rp} baseline (45.4% for *LLM Syntax* versus 66.9% for h^{rp}).

However, in other problems, such as *Hospital*, LLM-based predictions perform poorly and demonstrate significantly lower accuracy compared to h^{rp}. The best LLM heuristic in this problem, *LLM Natural with Limits*, only achieves an accuracy of 9.2% compared to the baseline's 49.6%. The LLM heuristics also had poor accuracy in predicting the length of plans for *Hospital* and *Jailbreak*. This suggests a correlation between a heuristic's ability to predict distance to the goal and to predict next actions.

One major issue in the *Hospital* domain is that, in the Sabre syntax versions, the LLM often misinterprets the structure of actions. It tends to treat actions as functions and tries to nest them incorrectly. For example, in the plan:

```
assess(Hathaway, Ross, symptom(Ross), PatientRoomB);
treat(Hathaway, Ross, treatment, PatientRoomB);
```

the LLM does not understand that `symptom(Ross)` is a fluent with a value rather than a function that can be nested. This misunderstanding leads to many incorrect plans. In the natural language versions, the problem is that the LLM often provides vague or incomplete actions. Instead of specifying treatment details, it generates actions like "Treat Jones" or "Treat Ross" without indicating the required treatment or location, such as "Treat Jones with Steroids at PatientRoomA." This lack of specificity significantly reduces accuracy.

In the *Lovers* domain, the poor performance appears to be due to the complexity of tracking multiple characters and objects, as well as their locations. Many actions require characters to be in the same room to interact, such as giving an item to another character. However, the LLM often forgets or disregards these spatial constraints, leading to unrealistic plans where characters exchange items or perform actions without being in the correct location. This suggests that the LLM struggles to maintain an accurate world state, which is crucial for effective planning in this domain.

In summary, while LLM-based heuristics offer competitive accuracy in select problems and demonstrate some potential as narrative planning heuristics, their performance is inconsistent. They excel in problems like *Space* and *Gramma*, yet struggle in problems like *Hospital*. This suggests that further refinement in prompt design and problem translation is necessary to improve the reliability of LLM-based heuristics across a broader range of problems.

Limitations

Our approach has several limitations. One significant limitation is the time it takes to call the GPT-4o mini API and receive a response from the LLM. This latency is currently a prohibitive bottleneck to any kind of online search, but we expect this to improve as LLMs become smaller and faster and once state-of-the-art language models can be run locally.

LLMs, including GPT-4o mini used in this study, can sometimes produce unpredictable outputs. This unpredictability can manifest in the length of the plan or the content, where the generated plans can be either excessively verbose or nonsensical.

Another limitation is related to prompt engineering. The effectiveness of the LLM's predictions depends havily on the quality of the prompts. The crafting of these prompts requires significant effort and expertise from the author and will be difficult for other scientists to replicate reliably. This process is also time-consuming and is prone to human error, which can impact the overall accuracy and efficiency of the approach.

Addressing these limitations will be crucial for improving the reliability and efficiency of using LLMs in narrative planning.

Conclusion

In this study, we evaluated the performance of LLM-based heuristics compared to traditional planning heuristics at completing partial stories. Our analysis revealed that, while LLM-based methods demonstrate potential in certain problems, they often fall short in consistently outperforming classical heuristics, especially in structured and complex problems like *Hospital* and *Jailbreak*. The results suggest that LLM-based approaches require careful prompt engineering and fine-tuning to optimize accuracy.

Fine-tuning ϵ values showed that specific adjustments could enhance prediction accuracy, but also highlighted the challenge of domain-specific optimization. LLM-based heuristics show varied performance across different problems, indicating that no single heuristic is universally effective for all narrative planning problems. This suggests a need for adaptive strategies that take into account problem characteristics.

Overall, our findings emphasize the potential of LLM-based heuristics, while also underscoring the importance of continued research in prompt design and hybrid heuristic methods to achieve consistent and reliable performance in narrative planning.

Code

To enable replication and to encourage other researchers to build on this work, we have made our prompts, evaluation scripts, and other materials available on GitHub, allowing others to repeat our experiments or test different LLMs.

https://github.com/lazzy07/llm_as_np_heuristic

Acknowledgments

This material is based upon work supported by the U.S. National Science Foundation under Grant No. 2145153 and the U.S. Army Research Office under Grant No. W911NF-24-1-0195. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Army Research Office.

References

[1] Avrim L. Blum and Merrick L. Furst. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90, 1 (1997), 281–300.
[2] Blai Bonet and Héctor Geffner. 2001. Planning as heuristic search. *Artificial Intelligence* 129, 1 (2001), 5–33.
[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165* (2020).
[4] Marc Cavazza, Fred Charles, and Steven J. Mead. 2002. Character-based interactive storytelling. *IEEE Intelligent Systems special issue on AI in Interactive Entertainment* 17, 4 (2002), 17–24.
[5] Alex J. Champandard, Tim Verweij, and Remco Straatman. 2009. The AI for Killzone 2's multiplayer bots. In *Proceedings of Game Developers Conference*.

[6] Markus Eger and Chris Martens. 2017. Practical specification of belief manipulation in games. In *Proceedings of the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 30–36.

[7] Rachelyn Farrell and Stephen G. Ware. 2024. Planning stories neurally. techrxiv:171085113.35202301

[8] Mira Fisher, Cory Siler, and Stephen G. Ware. 2022. Intelligent de-escalation training via emotion-inspired narrative planning. In *Proceedings of the 13th Intelligent Narrative Technologies workshop at the 18th AAAI international conference on Artificial Intelligence and Interactive Digital Entertainment*.

[9] Malte Helmert. 2006. New complexity results for classical planning benchmarks. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling*. 52–62.

[10] Jörg Hoffmann and Bernhard Nebel. 2001. The FF planning system: fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14 (2001), 253–302.

[11] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large Language Models are Zero-Shot Reasoners. arXiv:2205.11916 [cs.CL] https://arxiv.org/abs/2205.11916

[12] Michael Lebowitz. 1985. Story-telling as planning and learning. *Poetics* 14, 6 (1985), 483–502.

[13] Michael Mateas and Andrew Stern. 2005. Structuring content in the Façade interactive drama architecture. In *Proceedings of the 1st AAAI international conference on Artificial Intelligence and Interactive Digital Entertainment*. 93–98.

[14] James R. Meehan. 1977. TALE-SPIN, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*. 91–98.

[15] Jeff Orkin. 2003. Applying Goal-Oriented Action Planning to games. *AI Game Programming Wisdom* 2 (2003), 217–228.

[16] Edwin P. D. Pednault. 1994. ADL and the state-transition model of action. *Journal of Logic and Computation* 4, 5 (1994), 467–512.

[17] David Pizzi and Marc Cavazza. 2007. Affective storytelling based on characters' feelings. In *Proceedings of the AAAI Fall Symposium on Intelligent Narrative Technologies*. 111–118.

[18] Ira Pohl. 1970. First results on the effect of error in heuristic search. *Machine Intelligence* 5 (1970), 219–236.

[19] Julie Porteous, Marc Cavazza, and Fred Charles. 2010. Applying planning to interactive storytelling: Narrative control using state constraints. *ACM Transactions on Intelligent Systems and Technology* 1, 2 (2010), 1–21.

[20] Bram Ridder. 2001. *Improve Automated Game Testing Using Domain Independent AI Planning*. Keynote at the 17th AAAI conference on Artificial Intelligence and Interactive Digital Entertainment. https://www.youtube.com/watch?v=2KXmxuCjjCw

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

[22] Stuart J. Russell and Peter Norvig. 2021. *Artificial Intelligence: a modern approach, Capter 11* (4 ed.). Pearson.

[23] Rushit Sanghrajka, Robert Michael Young, and Brandon R. Thorne. 2022. HeadSpace: Incorporating Action Failure and Character Beliefs into Narrative Planning. In *Artificial Intelligence and Interactive Digital Entertainment Conference*. https://api.semanticscholar.org/CorpusID:251672437

[24] Jonathan Teutenberg and Julie Porteous. 2013. Efficient intent-based narrative generation using multiple planning agents. In *Proceedings of the 2013 international conference on Autonomous Agents and Multiagent Systems*. 603–610.

[25] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the Planning Abilities of Large Language Models - A Critical Investigation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[26] Stephen G. Ware and Rachelyn Farrell. 2023. *A Collection of Benchmark Problems for the Sabre Narrative Planner*. Technical Report. Narrative Intelligence Lab, University of Kentucky.

[27] Stephen G. Ware and Cory Siler. 2021. Sabre: A Narrative Planner Supporting Intention and Deep Theory of Mind. In *Proceedings of the 17th AAAI International Conference on Artificial Intelligence and Interactive Digital Entertainment*. 99–106.

[28] Stephen G. Ware and R. Michael Young. 2014. Glaive: a state-space narrative planner supporting intentionality and conflict. In *Proceedings of the 10th AAAI international conference on Artificial Intelligence and Interactive Digital Entertainment*. 80–86.

[29] Anbang Ye, Christopher Cui, Taiwei Shi, and Mark O. Riedl. 2022. Neural Story Planning. arXiv:2212.08718 https://arxiv.org/abs/2212.08718

[30] R. Michael Young. 1999. Notes on the use of plan structures in the creation of interactive plot. In *Proceedings of the AAAI Fall Symposium on Narrative Intelligence*. 164–167.

[31] R. Michael Young, Stephen G. Ware, Bradly A. Cassell, and Justus Robertson. 2013. Plans and planning in narrative generation: a review of plan-based approaches to the generation of story, discourse and interactivity in narratives. *Sprache und Datenverarbeitung, Special Issue on Formal and Computational Models of Narrative* 37, 1-2 (2013), 41–64.