Entertainment Computing 38 (2021) 100422

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: www.elsevier.com/locate/entcom

Let CONAN tell you a story: Procedural quest generation

Vincent Breault, Sébastien Ouellet, Jim Davies

Department of Cognitive Science, Carleton University, Ottawa K1S 5B6, Canada

A R T I C L E I N F O

Keywords:
Procedural generation
Video games
Narrative
Planning

A B S T R A C T

This work proposes an engine for the Creation Of Novel Adventure Narrative (CONAN), which is a procedural quest generator that uses a planning approach. The engine is tested on its ability to create quests, which are sets of actions that must be performed in order to achieve a certain goal. The engine takes in a world description represented as a set of facts, including characters, locations, and items, and generates quests according to the state of the world and the preferences of the characters in it. We evaluate quests through the classification of the motivations behind the quests. We also compare world descriptions of two different sizes. Compared to human structural quest classification, the current engine was found to be able to replicate the quest structures found in commercial video game quests.

1. Introduction

Procedural Content Generation for Games (PCG-G) is the use of computers algorithms to generate game content, determine if it is interesting, and select the best ones on behalf of the players [1].

This type of generation becomes quite useful in a consumer environment that is more and more demanding in terms of content. Game development costs are extremely high as the demand for highly complex games requires hundreds of person-hours of work in development studios. For instance, the Massively Multiplayer Online Role Playing Game (MMORPG) *World of Warcraft* has a total of 30,000 items, 5,300 creatures and 7600 quests, withs an estimated budget of twenty to one hundred and fifty million dollars for a single game [1]. Algorithms capable of offloading this task by automatically generating such content would be invaluable to the industry.

Many video games use a variety of different procedural generation (PG) techniques to generate content, such as graphics (e.g., textures in [2]) or vegetation, through such programs as SpeedTree [3]. The creation of environment and maps has also seen its share of procedural generation algorithms for the generation of buildings [4],[5], road networks [6] and maps [7,8]. PG can also be observed in the automatic or custom creation of more abstract elements such as ships or weaponry [9], Non-Player Character (NPC) behaviour or other such system behaviour [10].

1.1. Narrative generation and planning

The creation of narratives has traditionally been the domain of human authorship, be it for movies, music or video games. With advancement in computer technology and research, the creation of such content has seen a slight shift from the human authored to automatic computer generation. Using algorithms to procedurally create stories can effectively alleviate some of the burden from artists when creating a new piece.

Prince [11] defines narrative as “the recounting of a sequence of events that have a continuant subject and constitute a whole.” The causality of events in a story, that is, the relationship between the temporally ordered events that change the world state, is a property of narratives that ensures its coherence and a continuing subject [12].

The two main aspects of narrative are the logical causal progression of a plot, meaning that the events that occur in the narrative obey the rules of the world in which it takes place, and character believability, defined as the perception by the player that the characters act in a coherent fashion, without negatively influencing the suspension of disbelief [13]. This means that the events in the story must appear logical and the agents must appear intentional in order for a piece of narrative to make sense. Therefore, for a sequence of events to be considered a story, it must maintain coherent causal relationships between the events and feature actions that make characters believable.

In standard systems, stories are entirely designed by human authors. This limits the capabilities for adaptation to audience preferences and has very low value for repeated consumption of the media (replaying, rereading, or re-watching). In order to counteract this limitation, systems have been designed to generate stories either dynamically, as they unfold [14–16] or ahead of time [17–23], by selecting story elements, ordering them and presenting them to the audience [24–26].

While narratives are sequences of coherent and cohesive events that describe a series of changes in the world over the course of the story, plans consist of temporally ordered operations that transform the world state with each step. As such, plans are similar to stories in that both

https://doi.org/10.1016/j.entcom.2021.100422
Received 18 June 2019; Received in revised form 7 April 2020; Accepted 20 February 2021
Available online 25 February 2021

have ordering and causality of actions in a plan sequence [13]. Given the structural similarities between the outputs of AI planning agents and story structure, planning has seen a lot of use in the PG of stories [27]. In comparison to some other frameworks for narrative PG, such as story grammars or scripting, planning is attractive because it searches a much larger space of possible stories [27].

Planners require a domain theory containing all the possible events that can happen, an initial state (which, in the case of narrative planning, would be the story world and all that it contains), and a goal situation (which, in the case of narrative planning, is what the final state of the world has to be like for the goal to be considered achieved). The task of the planner is to find a sequence of actions that links the initial state to the goal state. In the context of narrative, planners are provided with an initial situation for a story, the events that unfold and the finale of the piece of narrative.

One of the reasons planners are attractive in the generation of interactive stories, such as those found in video games, is that interactive fiction requires an enormous cost in terms of content generation. The more choices the audience makes, the greater the combinatorial explosion of possible outcomes [19].

People have used planning and machine learning to modify previously human authored stories [28], to control character behaviour and the overarching story [18] or to generate fixed, step-by-step quests such as the ones found in an MMORPG [29].

Procedural generation of game narrative presents its own challenges. One of them is the fact that for classical planning algorithms, though they will successfully find a sequence of actions leading from the initial state to the goal state, their sequence is in no way guaranteed to make sense given the nature of the characters in the story. The planning problem itself does not concern itself with character believability [13]. This means that if the goal state has a princess locked up, there is nothing preventing traditional planning systems from having the princess lock herself up. Furthermore, many systems, such as [25], use human-authored stories or human authorial intent to be able to create a coherent and cohesive story. These systems, called deliberative narrative systems [13], often use centralized reasoning in order to produce a narrative that satisfies the constraints and parameters intended by the human authors.

Multi-agent simulation and distributed planning approaches create plans for each agent, depending on the current context and world state, solving the problem of making intentional agents [18]. In games, character believability is dependent on the coherent causal relationships between the character's attributes known to the player, such as the character's personality or desires, and said characters' actions. This means that when deciding upon sequences of actions, characters need to make plans according to their own goals in order to appear intentional [13]. By simulating a world with intentional characters, believable interactions can emerge from the simulations [30]. Many systems using emergence also use director agents in order to guide the story [31], satisfy author goals and ensure interesting and well-structured performance of the simulation.

1.2. Quests

In the context of video games, we have "quests," which are particular kinds of interactive narratives where audience/player choices control player characters in a way that affects the story [29]. Quests are multi-step tasks to achieve some goal, usually for a reward. They are often provided to player characters (PCs) by non-player characters (NPCs) within the game and are embedded in a piece of narrative that makes the sequence of actions make sense given the NPC and the current world state. One quest that Doran found has 27 steps, while others might have only a few [29].

[32] divides quests in three basic categories. The first is *place oriented*, where the player has to move their avatar through the world to reach a target location with puzzles along the way, such as in Cyan Inc.'s *Myst*. Slightly less common are the *time-oriented* quests, where the task of the quest might simply be to survive for an interval of time. Last is the *objective oriented* quest where the task is to achieve a certain objective, such as bringing an item somewhere or taking it by force from an agent in the world. These basic categories can be combined, nested and serialized to produce more or less complex quests.

The current work describes and evaluates our implemented system, the Creation Of Novel Adventure Narrative (CONAN). Our design goals are to have the NPCs be able to provide pertinent quests to the player, the sequence of which, in a persistent world, must be coherent. Assuming that human-authored quests are high-quality in these ways, the system must therefore be able to create similarly built quests to be deemed successful. This experiment aims to create the quest generation engine CONAN in such a way that the quests it creates and offers to the player are similar to the ones a human author would. Furthermore, the CONAN engine is designed to create quests make sense given the NPC communicating the quest in the game world.

2. CONAN design

In this project we have created a quest-generation engine called CONAN, implemented in software. The CONAN engine's goal is to produce as output novel and coherent quests in a video game context, using a planning algorithm. The CONAN engine accomplishes its goal by having the NPCs make plans to solve their goals in accordance with their preferences. The intention is that a game designer might use CONAN in a game to create quests for the player. As such, the designer would input the world and its NPCS, and CONAN can automatically generate quests at gametime to be presented to the player.

Additionally, because player action changes the world state, the system is able to endlessly produce more context-relevant quests as the simulation goes on. The initial world state is composed of locations, non-player characters with pre-defined preferences, monsters and items, laws governing the world (such as "when trees are cut down, there are fewer trees"), what actions are possible, as well as the prerequisites and results of said actions. Each of the objects are defined within the world state by statements such as `location(Castle)` pertaining to the castle object, defining it as a location. These specifications will determine what is possible within the world and thus which quests can be created.

These plans, which will now be referred to as quests, are intended to be transferred to the player character as requests. For instance, a baker NPC that ran out of bread might want to make more bread to sell, for which he needs more wheat. That NPC might then ask the player character to get him some wheat from the field in exchange for payment.

2.1. Initial State

CONAN has two 'users:' the game designer using CONAN to create a game, and player who plays the game. To avoid confusion, we will refer to these roles as 'designer' and 'player' rather than the ambiguous term 'user.'.

The initial state, input by the designer using CONAN, contains statements describing all that exists in the story world. We specifically use the modified Aladdin world that has seen much use in the literature [18,33,13] for comparison purposes. This means that the following elements are present:

- King Jafar, who lives in a Castle
- Aladdin, a knight who has a cooperative attitude towards King Jafar
- Jasmine, who also lives in The Castle
- A Genie, who is in the location 'Magic Lamp' and unable to get out
- A Dragon, who lives in the Mountain, is hostile to agents and guards the 'magic lamp'

The locations mentioned in the world description are interconnected such that one may move to and from the castle and the mountain. The Magic Lamp is only accessible from the cave.

Each of the NPCs are associated with preferences represented by values weighting the actions, described in the domain section (see Table 1). Part of what makes a given quest seem plausible is if it makes sense given the personality of the NPC providing it. A good character should not ask the player to murder an innocent person, for example. To make quests psychologically realistic, we introduce a novel method: each NPC has specific costs associated with each action. In planning, each action can have a cost associated with it, and the planning algorithm tries to find an efficient plan to the goal given the costs. Aladdin might then have a lower cost to the 'Defend' action. These will be used by the planning system to find goals and sequences of actions for each agent such that they conform to the NPC's action preferences. If different NPCs have different costs associated with each action, then even if the goal is the same, the planner will generate different quests for each NPC.

To test the system on how well it scales to something larger, we also ran CONAN on a second, more complex initial world state, which we will refer to as "Medieval Valley:"

- Agents
  - Baker
  - King
  - Lumberjack
  - Blacksmith
  - Merchant
  - Guard
  - Daughter
- Locations:
  - The Castle, connected to the village
  - The Village, connected to the castle, the bakery, the shop, the wheat field and the forest
  - The Wheat field, connected to the village.
  - The Cave, connected to the forest.
  - The Bakery, connected to the village
  - The Forge, connected to the cave
  - The Forest, connected to the village and the cave
  - The Shop, connected to the village

The items in this world are a hammer, wheat, a sword and a magic spell book. The monsters (the troll, the wolves and the slimes) are considered to be items for implementation purposes.

2.2. The domain file

Doran and Parberry conducted an analysis of several popular MMORPGs and generated a list of basic actions that describe the steps involved with most quests [24]. The actions are as follows: DoNothing, Capture, Damage, Defend, Escort, Exchange, Experiment, Explore, Gather, Give, GoTo, Kill, Listen, Read, Repair, Report, Spy, Stealth, Take and Use. We will assume this set of actions adequately describes the kinds of actions needed to be undertaken to complete human-authored quests, and as such they are the actions we implemented in CONAN. This is so that the generated quests can include all possible actions and resemble human created quests.

Each action will be implemented in the system the following way, using PDDL syntax, which is the standard for planning algorithms [34]:

```
(:action move
:parameters (?p?to?from)
:precondition (and (location?to)
(location?from)
(player?p) (at?p?from))
:effect (and (at?p?to)
(not (at?p?from))
(increase (total-cost) 2)))
```

Each action thus contains a name, the parameters used (player, destination and current location in the case of the above example), preconditions establishing what conditions must be true in order for the action to be available (such as the destination being a location) and what the effect of said action will have on the world once it has been performed. Additionally, actions are weighted in accordance with NPC preferences, with actions incompatible with the NPC's preference having higher cost than those in line with the preferences. This is represented with the `increase (total-cost)` part of the effect, with the number representing the NPC's preference for this specific action. All actions have a base cost of 2.

This means that an action such as Kill will have a path cost higher (raise to a cost of 3) for the baker than the knight, for instance. These mappings of actions to preferences of the NPCs will guarantee that the NPC's choice of actions will be coherent with its personality The action preferences of the agents are described in Table 1.

In Table 1, actions preceded by a '-' have a lower cost (and are therefore preferred) and those with a + sign have a higher cost, making them less desirable. In our example each character has been given four or five actions with differing costs, though any number of actions can be modified in this way in order to change the NPC's preference, and in turn the resulting kinds of quests they might give to players. For instance, Jafar has a preference for the actions 'kill', 'spy', 'take', 'stealth', and 'move'. This will cause him to prefer plans that involve sneaky actions, murder and stealing, as an emergent property. This ensures that different NPCs will create different quests by giving them a semblance of personality. Similarly, the Dragon has -kill, so its preferences-defined personality makes it more likely to recommend killing an adversary than dealing with problems in a more peaceful way. In contrast a law-abiding citizen would recommend a significantly different quest for a similar objective.

Currently there is no automated mechanism that makes sure these NPC personalities make sense. For now, this is up to the designer using the system to have the action preferences reflect those of plausible persons.

2.3. Goal generation

The goals themselves take the form of sets of statements that must become true in the world state. These statements are a combination of predicates, such as 'has' or 'defended', and an object, such as an agent, an item, or a location. For example, one statement in a set might be `(has baker wheat)`, meaning that in order for the goal to be accomplished, that statement has to become true. In the current implementation of the program the possible predicates are as variables: `?i` being information, `?l` a location, `?o` an object, `?c` a character, `?p` the player, and `?cl` a character at a location:

```
predicates = [
"(has?cl?i)",
"(has?p?o)",
"(has?c?o)",
"(cooperative?c)",
"(at?c?l)",
"(character?c)",
"(captive?p?c)",
"(damaged?i)",
"(defended?c)",
```

(continued on next page)

Table 1
Aladdin World character preferences represented with action cost modifiers.

Character	Preferences Represented as Costs
Aladdin	["+kill","−exchange","−use","+escort"]
Dragon	["−damage","−take","−report","+escort","−defend"]
Genie	["−kill", "−exchange", "−defend", "−read"]
Jasmine	["+kill", "−spy", "−take", "−stealth"]
Jafar	["−kill", "−spy", "−take", "−stealth", "−move"]

(*continued*)

```
"(defended?i)",
"(sneaky?p)",
"(dead?m)",
"(experimented?i)",
"(explored?l)",
"(used?i)"]
```

This representation restricts what kinds of characters and entities can be involved with any particular predicate. For example, the player can only make herself 'sneaky' as the 'sneaky' predicate can only be attributed to ?p, the player.

To make a given NPC's goals psychologically realistic, CONAN gives them goals that match their preferences. First, CONAN chooses a (designer-defined) number of random predicates from the above list. It then assigns a random item in place of each ? object. It generates plans to achieve these goals. The average cost of the actions in the plan is calculated (using that particular NPC's action preferences). Given that the cost for actions in line with the preference of an agent is 1, the closer the average is to 1 the more the plan is in line with the character. This is repeated for each goal state and the one yielding the plan with the lowest cost is kept. In our test, CONAN creates four random goal states to be evaluated in this fashion for each character. Having more random goal states means the system is more likely to find a better goal but has the cost of being more taxing in terms of computation. The spline curve in Fig. 1 shows how the mean of action costs within a plan changes with the number of goal states attempted (see Figs. 2 and 3).

To see if this works, we also implemented a random goal generator for the purposes of comparison.

2.4. Simulation

CONAN is part of a simulation that runs in a turn-based manner. There are two turns: the NPC turn and the PC turn.

In the NPC turn, each NPC designs their quests (represented as plans) to give to the player. This means that CONAN, after taking in the designer-defined initial state and domain files with the actions, will solve each NPC's planning problem and present the player with each plan. Before it is given to the player, a distribution of the quest motivations for the generated quests is computed for evaluation purposes. A translation module then transforms the plans from the series of action names to a readable form for the player.

Next is the player turn, where the player may perform an action that impacts the game world. It is the opportunity for the player to interact with the NPC–that is, to help them with the quest they have transferred onto him. Once the action is taken, the world states are updated and the system cycles back to the NPC turn, generating new plans in light of the changes that have occurred in the world.

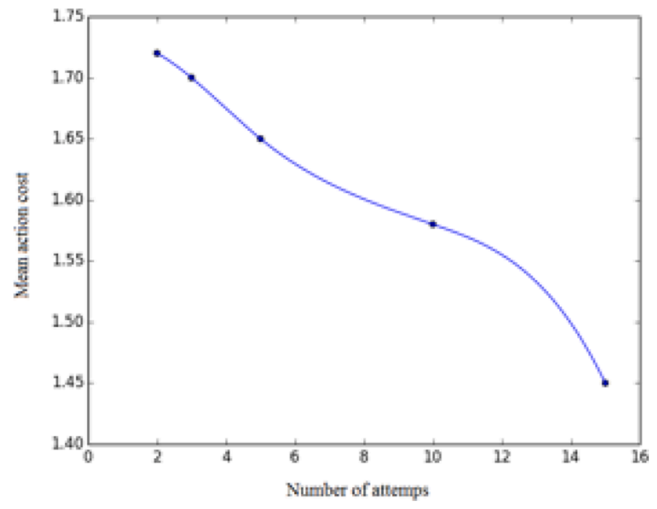

Fig. 1. Mean of action cost by number of goal states attempted. It shows a downward trend of the discrepancy between the generated quest and the NPC's preferences based on the number of iterations before settling on a quest.

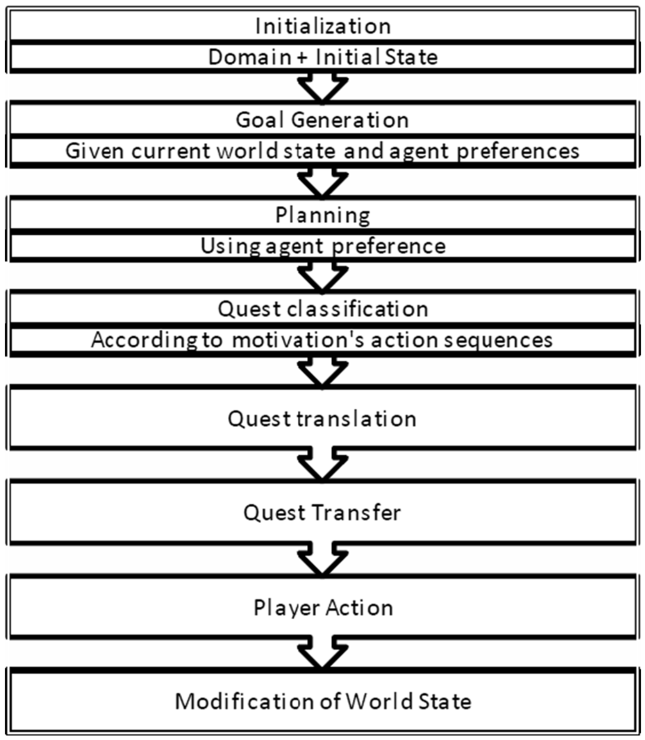

Fig. 2. Flow of operation during the course of an NPC turn and a PC turn. The simulation repeats these steps.

2.5. Quests

As mentioned previously, the CONAN engine presents the player with various quests at each time step. A "quest" is represented as a series of steps generated by the planner, which must be taken in order to solve some problem. The quests are formally and structurally defined as follows.

Doran and Parberry's analysis of more than 3000 human authored quests designed for commercial video games classified quests into nine categories, described as underlying 'motivation' [29,24]: knowledge, comfort, reputation, serenity, protection, conquest, wealth, ability, and equipment. For example, a quest from a baker wanting more wheat would fall under the "Wealth" category.

Table 2 describes examples of quests from the "Medieval Valley" world that the CONAN engine could generate for each of the different categories. We take Doran and Parberry's ontology as representing the breadth of quest types that appear in human-generated content. We will consider CONAN's quests to be similar to human-generated quests (and therefore of some quality) to the extent to which CONAN creates quests in more of these categories.

Furthermore, given the actions that exist within the current software and its implementation, the quests will fall within the first and last categories of the tree, as described by [32], 'place oriented' and 'objective oriented.' The second category, time-oriented, cannot exist in the current version as there is no implemented concept of duration.

We expect that smaller, simpler initial world states will create fewer, less varied quests while more complex worlds, which we expect will create quests spanning all quest motivations. In the context of this evaluation, we predict that the Aladdin world will generate quests in

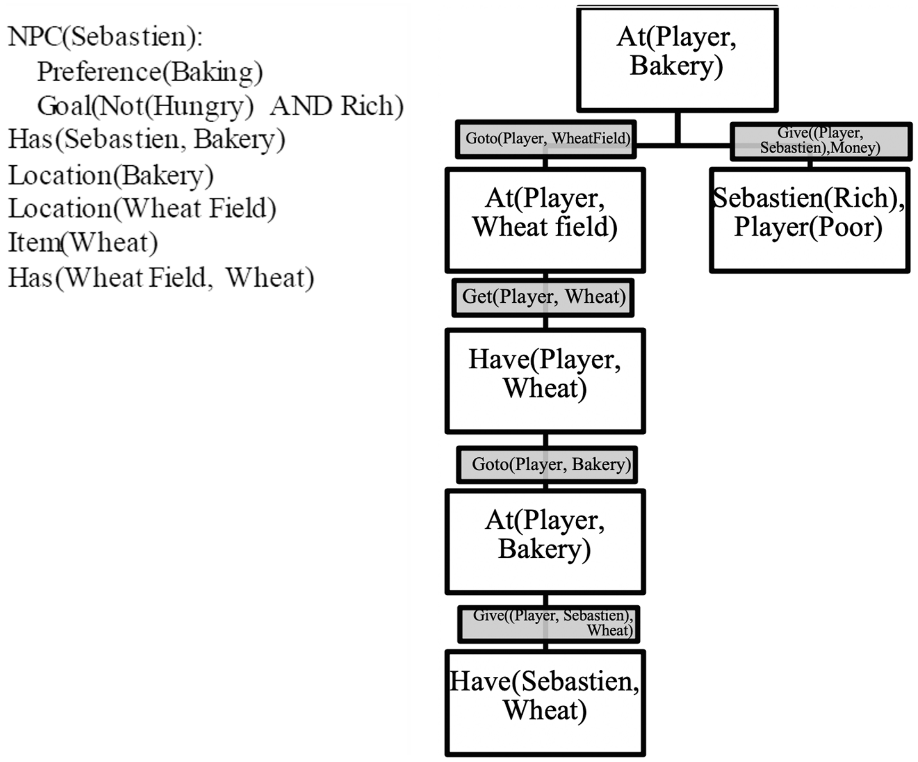

Fig. 3. Sample world state and plan for solving the need for wheat from the baker. Both branches show possible plans solving the goals of the NPC.

Table 2
Categories of Quest Motivation from [24].

Motivation	Example of quest
Knowledge	"Find the location of the king's stolen treasure"
Comfort	"Get rid of the wolves in the forest that are preventing the lumberjack from getting wood."
Reputation	"Get granite and build a statue of me in the town square."
Serenity	"Rescue the daughter of the baker that was taken by a troll."
Protection	"Go kill the troll that has been traumatizing the village"
Conquest	"Go kill my enemies."
Wealth	"Go get some wood for the lumberjack to sell."
Ability	"Find me the ancient spell book."
Equipment	"Assemble the lumberjacks' axe."

fewer categories than those generated by the larger Medieval Valley world.

The solution to each NPC's current problem or objective is processed individually by a planning agent. In order to do that, each NPC is instantiated as a planning problem with its own set of constraints and goals and its own domain. As [18] have noted, although authorial intent suffers from such distributed processing, characters should be more realistic and believable, which had been noted as providing meaningful interaction [30,18].

2.6. CONAN's planning algorithm

The CONAN engine is designed so that NPC intentionality results from the different preferences, as represented as differing action costs for the planner. More complex distributed algorithms, such as MAPL [17], are unnecessarily complex for our purposes of handling of intentionality by the global planner [13,18]. The CONAN system uses Fast Downward [35], as distributed under the GNU General Public License. It is an implementation of a classical planning system based on heuristic search. It is encoded in PDDL 2.2, which is a standardized format for planning, and supports features such as ADL conditions and effect. This had the advantage of allowing numerical values to be used in the domain file, which we use here for weight of each action to be added in the plan weight. ADL also provides support for negative statements in preconditions. The planner supports many heuristics; we used A* in the current implementation. It takes in a standard PDDL format problem, translates it into multi-valued planning task and uses a hierarchical decomposition of the planning task to compute the heuristic function. Its use in other work has proven it to be a very efficient implementation [35].

In deployed games, often simple quests are given first, followed by longer and more challenging quests. The current implementation of CONAN does not support this, and this addition reserved for future work.

2.7. Interactivity

Although CONAN is not implemented in a proper game, it has a rudimentary text interface that shows a player a suggested quest. The player can input actions to complete quests. The next quest given to the player is a function of the new world state as affected by player action. As will be discussed in the Discussion section, our theory is that a compelling story will emerge from player choices, though we leave evaluation of this theory to future work.

3. Methods and evaluation

We wanted to test whether CONAN could make the variety of quest types that have been observed in the underlying motivations found in human-authored quests. To do this, we had CONAN generate a large number of quests and classified them. If all of the quest motivations were well-represented, then we would consider CONAN to be successful in this regard.

For purposes of this evaluation, we assume the quest is no more than the plan output by CONAN. Each generated plan consists of a sequence of actions. For classification, we ignore variable values and only look at the abstract action. `get(sword)`, for instance, would only be treated as a generic `get` action.

Doran and Parberry provide tables describing which action sequences belong to particular strategies, which in turn are associated with the nine underlying motivations [24]. As seen in Table 3, a plan like `get, goto, give` has a sequence of actions that match the *Deliver item for study* strategy, which belongs to the *Knowledge* motivation.[1](see Table 4).

Each plan that CONAN generates might involve sub-sequences of actions that match more than one motivation. For each strategy found in the quest, its associated motivation will be given a score and the motivation with the highest score will be determined as the overall motivation for that particular quest. We will consider CONAN to be exhaustive in its breadth of possible quests if it is able to cover all the nine motivations that were found underlying all the human authored quests.

3.1. Classifier module

In order to determine the motivation category of each of the created quests, CONAN has a built-in classification module. Table 2 shows the motivation categories Doran and Parberry [24] found in the quests of commercial MMORPGs.

Doran and Parberry's [24] ontology associates certain sequences of action with strategies that are uniquely associated with motivations. Each motivation has between two and seven unique strategies. The complete list can be found in [24]. The motivations and strategies are descriptions in English, and the sequences of actions are lists of atomic actions, such as <get> <goto> <give>, as shown in Table 3. When we refer to a "quest" in this context we mean the plan output by CONAN, and when we refer to a "sequence of actions" we mean a list of actions that [24] associates with strategies and motivations.

For example, the quest motivation "Knowledge" has four associated strategies, which are unique to it. These are "Deliver item for study", "Spy", "Interview NPC", and "Use an item in the field". Each of these strategies is described by a particular (not not unique) sequence of actions, an example of which is shown in Table 3.

The classifier, in order to determine to which motivation a given generated quest belongs to, creates a fuzzy membership-based classification where each quest belongs to a greater or lesser degree (0 to 1) to every motivation. After the quest is fully analyzed, the overall motivation for that particular quest is assumed to be that which has the highest level of membership.

To do this, the quests are segmented into pairs of actions, each of which the classifier compares to every strategy in every motivation, and looks for a match. For example, if the quest is <get> <goto> <give>, then it will have two pairs: <get> <goto > and <goto> <give>. These pairs are compared to the sequences of actions that are associated with particular strategies. If there is a match, then the membership of that quest in a particular motivation (the one associated with that strategy) is augmented.

The motivation with the highest score is determined to be the overall motivation for the quest, as described by the following:

$$\underset{m \in \mathscr{M}}{\operatorname{argmax}} \sum_{i}^{N} \frac{1}{L_i \times N}$$

Where m is the current motivation among the set of motivations $\mathscr{M}$, N is the number of strategies in that motivation, and L_i is the length of a given action sequence, specifically the number of pairs of actions. Because different strategies are associated with different lengths of action sequences, this equation normalizes it. This method of classifying quests has the advantage of allowing the classifier to detect strategies even if they are not complete, as long as the sequence of actions is closer to one given strategy than another.

Since some strategies are comprised of only one action, such as the 'Ability' motivation with its 'Use', and 'Damage' strategies, the classifier will consider a sequence of actions such as ('Move' 'Damage') to be both from 'Ability' because of the 'Damage' and from the 'Serenity' motivation, as one of its strategies is ('Move' 'Damage'). This causes the motivation with single action strategies to be over represented, such as 'Ability'.

Another limitation is that the classifier also does not account for possible sub-quests or embedded quests as having distinct motivations. For instance, if the quest was to destroy something with an axe but the character first had to get said axe, the classifier would count this as only one quest with one motivation instead of a quest with an embedded sub-quest.

3.1.1. Classifier module evaluation

In order to evaluate the classifier, its output was compared to motivations assigned to quests by two of the authors.

We took the first 50 quests output by the "Medieval Valley" large-world test. Here are ten of the quests generated:

1. (stealth you) (move you village bakery) (take you guard sword village) (giveto you guard sword village) (take you guard sword village) (move you field village) (damage you wheat field sword)
2. (stealth you) (move you village bakery) (take you guard sword village) (experiment you sword) (move you castle village) (spy you king castle secret)
3. (explore you bakery forge) (escort you blacksmith forge castle) (listen you king castle secret) (report you blacksmith secret castle) (getfromlocation you castle spellbook) (escort you blacksmith castle village) (exchange you guard sword spellbook village) (escort you blacksmith village field) (kill you slime gel field sword)
4. (stealth you) (move you cave bakery) (take you troll hammer cave) (use you hammer)

Table 3
Strategies and sequences of actions for the Knowledge motivation, from [24].

Motivation	Strategy	Sequence of action
Knowledge	Deliver item for study	< get> < goto> < give>
	Spy	< spy>
	Interview NPC	< goto> < listen> < goto> < report >
	Use an item in the field	< get> < goto> < use > < goto> < give>

Table 4
10 of the 50 quests generated for classification, and their classifications from the classifier module and two of the authors. The numbers in the Quest column refer to the quests in the enumerated list of example generated quests listed above.

Quest	Classifier	Author1	Author2
1	Conquest	Conquest	Conquest
2	Knowledge	Knowledge	Knowledge
3	Not found	Protection	Reputation
4	Ability	Ability	Ability
5	Reputation	Protection	Protection
6	Serenity	Protection	Reputation
7	Protection	Protection	Protection
8	Reputation	Reputation	Equipment
9	Not found	Serenity	Conquest
10	Not found	Protection	Reputation

[1] Doran and Parberry use two slightly different ontologies in their two papers. For example, *Spy* is a strategy in [29] but in [24] it is called *Observing*. We made a adjustments to accommodate these minor differences.

5. (stealth you) (move you cave bakery) (take you troll hammer cave) (move you field cave) (kill you slime gel field hammer) (defend you gel field)
6. (move you castle bakery) (listen you king castle secret) (move you cave castle) (stealth you) (take you troll hammer cave) (move you village cave) (take you guard sword village) (giveto you guard hammer village) (escort you guard village field) (kill you guard hammer field sword)
7. (move you field bakery) (defend you wheat field)
8. (stealth you) (move you cave bakery) (take you troll hammer cave) (move you forest cave) (kill you wolf pelt forest hammer) (move you castle forest) (giveto you king hammer castle)
9. (stealth you) (move you village bakery) (capture you guard village)
10. (stealth you) (explore you bakery village) (take you guard sword village) (explore you village cave) (kill you troll hammer cave sword) (defend you hammer cave)

These 50 quests were classified by two of the authors into each of the 9 motivation categories. This was done not by implementing the classifier module's algorithm by hand, but rather using our common-sense notions of what kinds of quest belong to which motivations. The classifier module then classified the same 50 quests.

Out of the 50 motivation assignments, 14 were not identified by the classifier as belonging to any motivation.

There was generally mild agreement between the two raters and the automated classfier. The inter-rater agreement between the two humans is 0.44, as measured by Krippendorff's alpha, where 1 indicates perfect reliability, 0 indicates no agreement at all, and a negative number represents systematic disagreement [36]. The agreements for each human paired with the classifier is 0.40 and 0.38, and the agreement for the three together is 0.42. While the agreement is generally mild, the results are close to each other. This may indicate that the classification task is hard to complete, and that two humans and the classifier agreed at a similar level.

The differences in classifications by the classifier module and the authors can be explained in part by the inherent nature of the quest structure found in [29]. Although the motivations have specific strategies, some of them are very similar. For instance, the Knowledge motivation has the "deliver item for study" and the Equipment motivation has the "deliver supplies" strategy. Both these strategies have for sequence of action "<get> <goto> <give> ". Such similarities lead to ambiguity in decision of which motivation should be assigned to a given quest.

If we look at quest 5, for example, we can see that the module classified it as *reputation* but both authors classified it as *protection.* This, perhaps, because the final act in the quest is to defend something. It could be that people's interpretations of quests weigh the final action a bit more heavily, as the other actions might be seen as preparatory for it. A future version of the classifier might explore this possibility.

3.2. Large World Test: Medieval Valley

In order to evaluate the breadth of possible quests, the simulation needed to create many quests. CONAN used our larger world, called "Medieval Valley," to create about a thousand quests per NPC. The goals were randomly chosen. The quest actions were influenced by NPC preference.

The classifier module classified each quest into one of the 9 motivations found in [24]. Fig. 4 shows the resulting motivation distribution.

As one may see from Fig. 4, all the motivations were found in the Large World Test. Using a single initial state, CONAN created quests spanning all of the broad categories describing human-authored quests. This confirms that the engine is indeed capable of creating a wide variety of quests to present to the player audience. The distribution in Fig. 4 appears to be different from the distribution of human-generated quests seen in Fig. 5, as described in [24]. We speculate that a possible explanation for this difference can be attributed to human author preference and current popularity in the game market with respect to some kinds of motivations over others.

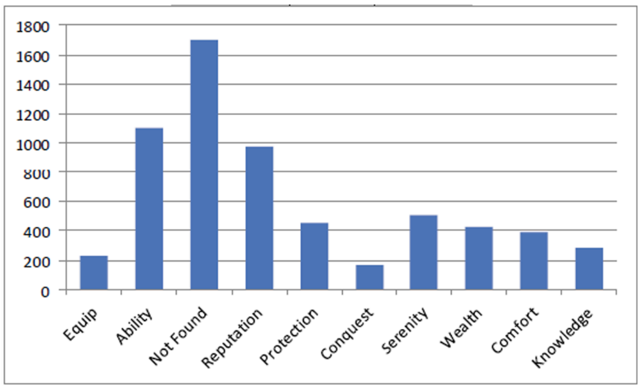

Fig. 4. Distribution of quest motivations as classified by the classifier module for the Large World "Medieval Valley" Test. All types of quest categories are accounted for in the Large World Test.

Many quests (1264) fell under the 'Not Found' category. These are quests that were either impossible to execute or already complete within the world. Given that for this test, the goals were chosen randomly, CONAN chose some goals that were already true in the world, such as `(alive baker)`. These goals could not produce quests and were therefore placed under the 'Not Found' category. Similarly, quests for which a successful plan could not be found within a time threshold (5 min) of searching were also dismissed as 'Not Found'. We would consider these to be unplayable quests.

3.3. Aladdin world test

The second test used the modified Aladdin world for its simulation. This world is much simpler, having fewer facts and objects than the Large World. The same test was performed, with random goals and about a thousand quests for each of the 5 agents. The results are seen in Fig. 6. Given its much simpler nature, the world did not produce the same variety of quests as the Large World. This suggests that although CONAN can produce quests in all motivations, it requires a world of sufficient complexity to do so.

4. Discussion

In conclusion, the CONAN engine is indeed capable of creating quests

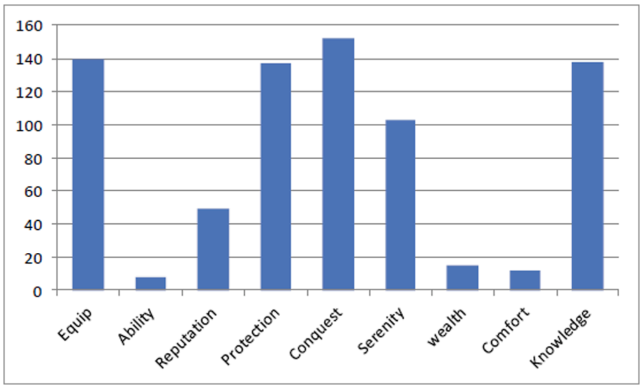

Fig. 5. Distribution of motivation in human written quests, as reported by Doran and Parberry in [24] . While all motivations are found, the distribution is different from what we have seen in quests generated by the engine in the Large World Test.

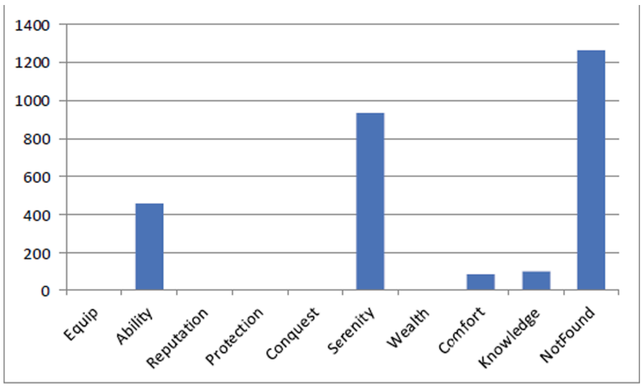

Fig. 6. Distribution of quest motivations in the Aladdin World, as classified by the classifier module. We observe that a more limited world, with a smaller set of facts and characters, produces a narrower range of quest motivations.

that resemble human written quests, as classified by the underlying motivation of quests. Although the tests showed a different motivation distribution for the sets of quests found from the human written quests, the engine can indeed create quests that cover the entire set of categories. It has also shown that although worlds with few objects could not produce all possible quest motivation, the CONAN engine can produce them given a complex enough initial world state, supporting the claim that the complexity of the produced quests depends in part on the complexity of the world given as input.

4.1. Classifier module

One of the contributions of this work is the classifier module. Evaluating creative systems are expensive, as they often require panels of independent expert raters to judge the quality of the output. Although our classifier does not measure the overall quality of the quests (for example, in terms of creativity or how much fun they would be), this work offers an objective evaluation that does not require human raters. The classifier module automatically determines the range of quest motivations, speaking to the breadth of a system's output. The code for the classifier, as well as for CONAN, can be found at https://github.com/science-of-imagination/conan-procedural-quest-generation.

CONAN itself differs significantly from other systems in three ways.

4.2. Psychological Plausible Quests Using Action Preference

CONAN explicitly represents NPC action preferences in an effort to make the quests they give more psychologically realistic. It does this by changing the cost of each action in the planning system, a method that has not been done before. Though we have not tested whether this was effective or not, other systems, such as CiF-CK and *Comme il Faut*, specifically target character believablity, making this a promising future direction [23]. Riedl and Young's planner [13] uses character traits to help shape plot, but does not do it at the level of preferences for particular actions.

4.3. Using planning to generate quests

Second, it generates quests by using planning. There are many systems that use planning to generate plots, but not specifically quests [27], such as Stefnisson and Thue's Mimisbrunnur [19], Ware and Young's Glaive [20] and COCPL [21], Cardona-Rivera and Young's INDEXTER [22] Riedl and Young's planner [13], and a planner by Thue et al. [37]. State of the art systems such as IMPRACTical [18] seek to create stories through intentional planning by multiple agents and a single narrative planner that generates the narrative story. Brenner's MAPSIM [17] uses continuous multi-agent planning in order to write a story with its different agent's goals and intentions, and Riedl, Thue, and Bulitko's work [25] uses a centralized planner to adapt plotlines in order to create new stories.

The only quest-generation system we know of is Doran and Parberry's [29], which uses structural *rules* to create quests. In contrast, CONAN uses planning. We found in our study that CONAN can generate quests of every kind of motivation if given a complex enough input world. Interestingly, the standard Aladdin world used by many systems does not seem to be large enough to generate the range of motivations found in human-authored quests.

4.4. Interaction and the Emergence of Story

CONAN has a rudimentary player interaction interface. After every player action, quests are reevaluated with the changes that the player made to the world. This allows the world to change over time and generate quests not possible before the player takes action. The set of possible quests changes as the simulation goes on, because different facts in the world state come in and out of existence, effectively creating an limitless changing set of possible quests.

We speculate that this interaction will create an overall story. It could be that a coherent story will emerge from the human player, through their interactions in the game, might act as their own director agent, forming a coherent narrative through their own choice of quests. Evaluation of this part of CONAN is reserved for future work.

This is contrast with some systems, which use story planners or director agents. In director agent systems [31,38], the AI entity chooses interesting plot points and adjusts the story to maximize its entertainment value. We suggest here that the player's choice of which quest to undertake within a large set of available quests, which sub-story to pursue, and which actions to perform in the world will ensure that the story elements, the quests as they unfold, will be interesting to the player.

Future work should investigate the effectiveness of the interaction aspect of this idea. Specifically, one might look at subjective experience of player reporting on both their interactions with the characters in the world but also of their experience of any single simulation's story as they progress through the quest and interact with the agents in the world, giving information on the believability of the characters. Preliminary work on this is represented in the PaSSAGE system [14], under the Delayed Authoring Theory [15], and the StoryAssembler system [16].

An engine like CONAN, effectively capable of autonomously creating infinity of quests would be valuable to an industry that is growing in demand for complexity and which uses human artists and authors to create such content. The possibility to offload this task, partially or totally onto an autonomous system would make the creation of such games much cheaper and create a very high replay value as the gameplay would be different every time the game is played by its intended audience.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary material

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.entcom.2021.100422.

References

[1] M. Hendrikx, S. Meijer, J. Van Der Velden, A. Iosup, Procedural content generation for games: A survey, ACM Trans. Multimedia Comput., Commun., Appl. (TOMM) 9 (1) (2013) 1.

[2] S. Lefebvre, F. Neyret, Pattern based procedural textures, in: Proceedings of the 2003 symposium on Interactive 3D graphics, ACM, 2003, pp. 203–212.
[3] A. de la Re, F. Abad, E. Camahort, M.C. Juan, Tools for procedural generation of plants in virtual scenes, in: Computational Science–ICCS 2009, Springer, 2009, pp. 801–810.
[4] A. Martin, A. Lim, S. Colton, C. Browne, Evolving 3d buildings for the prototype video game subversion, in: European Conference on the Applications of Evolutionary Computation, Springer, 2010, pp. 111–120.
[5] G. Kelly, H. McCabe, Citygen: An interactive system for procedural city generation, in: Fifth International Conference on Game Design and Technology, 2007, pp. 8–16.
[6] J. Sun, X. Yu, G. Baciu, M. Green, Template-based generation of road networks for virtual city modeling, in, in: Proceedings of the ACM symposium on Virtual reality software and technology, ACM, 2002, pp. 33–40.
[7] K. Hartsook, A. Zook, S. Das, M.O. Riedl, Toward supporting stories with procedurally generated game worlds, in: 2011 IEEE Conference on Computational Intelligence and Games (CIG'11), IEEE, 2011, pp. 297–304.
[8] J. Togelius, M. Preuss, G.N. Yannakakis, Towards multiobjective procedural map generation, in: Proceedings of the 2010 workshop on procedural content generation in games, ACM, 2010, p. 3.
[9] E.J. Hastings, R.K. Guha, K.O. Stanley, Automatic content generation in the galactic arms race video game, IEEE Trans. Comput. Intell. AI in Games 1 (4) (2009) 245–263.
[10] J. Orkin, Agent architecture considerations for real-time planning in games, AIIDE (2005) 105–110.
[11] G. Prince, A dictionary of narratology, U of, Nebraska Press, 2003.
[12] S. Chatman, Reading narrative fiction, 1993.
[13] M.O. Riedl, R.M. Young, Narrative planning: balancing plot and character, J. Artif. Intell. Res. 39 (1) (2010) 217–268.
[14] D. Thue, V. Bulitko, M. Spetch, Passage: A demonstration of player modeling in interactive storytelling, AIIDE 8 (2008) 227–228.
[15] D. Thue, V. Bulitko, M. Spetch, Making stories player-specific: Delayed authoring in interactive storytelling, in: Joint International Conference on Interactive Digital Storytelling, Springer, 2008, pp. 230–241.
[16] J. Garbe, M. Kreminski, B. Samuel, N. Wardrip-Fruin, M. Mateas, Storyassembler: an engine for generating dynamic choice-driven narratives, in, in: Proceedings of the 14th International Conference on the Foundations of Digital Games, 2019, pp. 1–10.
[17] M. Brenner, Creating dynamic story plots with continual multiagent planning., in, in: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, 2010.
[18] J. Teutenberg, J. Porteous, Efficient intent-based narrative generation using multiple planning agents, in: Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 603–610.
[19] I.S. Stefnisson, D. Thue, Mimisbrunnur: ai-assisted authoring for interactive storytelling, in: Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference, 2018.
[20] S.G. Ware, R.M. Young, Glaive: a state-space narrative planner supporting intentionality and conflict, in: Tenth Artificial Intelligence and Interactive Digital Entertainment Conference, 2014.
[21] S.G. Ware, R.M. Young, Cpocl: A narrative planner supporting conflict, in: Seventh artificial intelligence and interactive digital entertainment conference, 2011.
[22] R.E. Cardona-Rivera, R.M. Young, A knowledge representation that models memory in narrative comprehension, in: Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.
[23] M. Guimaraes, P. Santos, A. Jhala, Cif-ck: An architecture for social npcs in commercial games, in: 2017 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, 2017, pp. 126–133.
[24] J. Doran, I. Parberry, A prototype quest generator based on a structural analysis of quests from four mmorpgs, in: Proceedings of the 2nd international workshop on procedural content generation in games, ACM, 2011, p. 1.
[25] M. Riedl, D. Thue, V. Bulitko, Game ai as storytelling, in: Artificial Intelligence for Computer Games, Springer, 2011, pp. 125–150.
[26] H. Yu, M.O. Riedl, A sequential recommendation approach for interactive personalized story generation, in, in: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 71–78.
[27] R.M. Young, S.G. Ware, B.A. Cassell, J. Robertson, 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.
[28] B. Li, M.O. Riedl, An offline planning approach to game plotline adaptation, in: AIIDE, 2010.
[29] J. Doran, I. Parberry, Towards procedural quest generation: A structural analysis of rpg quests, Dept. Comput. Sci. Eng., Univ. North Texas, Tech. Rep. LARC-2010 2 (2010).
[30] R.S. Aylett, S. Louchart, J. Dias, A. Paiva, M. Vala, Fearnot!–an experiment in emergent narrative, in: International Workshop on Intelligent Virtual Agents, , Springer, 2005, pp. 305–316.
[31] B. Magerko, J. Laird, M. Assanie, A. Kerfoot, D. Stokes, Ai characters and directors for interactive computer games, Ann Arbor 1001 (48) (2004) 109–2110.
[32] E. Aarseth, From hunt the wumpus to everquest: introduction to quest theory, in: International Conference on Entertainment Computing, Springer, 2005, pp. 496–506.
[33] P. Haslum, Narrative planning: Compilations to classical planning, J. Artif. Intell. Res. (2012).
[34] S. Russell, P. Norvig, A. Intelligence, A modern approach, Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs 25 (1995).
[35] M. Helmert, The fast downward planning system, J. Artif. Intell. Res. (JAIR) 26 (2006) 191–246.
[36] A.F. Hayes, K. Krippendorff, Answering the call for a standard reliability measure for coding data, Commun. Methods Measures 1 (1) (2007) 77–89.
[37] D. Thue, S. Schiffel, T. Gumundsson, G.F. Kristjánsson, K. Eiríksson, M. V. Björnsson, Open world story generation for increased expressive range, in: International Conference on Interactive Digital Storytelling, Springer, 2017, pp. 313–316.
[38] J. Laird, M. VanLent, Human-level ai's killer application: Interactive computer games, AI Magaz. 22 (2) (2001) 15.