Adaptive Branching Quests Based on Automated Planning and Story Arcs

Edirlei Soares de Lima
Faculty of Design, Technology and Communication – IADE
Universidade Europeia
Lisbon, Portugal
edirlei.lima@universidadeeuropeia.pt

Bruno Feijó
Department of Informatics
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
bfeijo@inf.puc-rio.br

Antonio L. Furtado
Department of Informatics
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
furtado@inf.puc-rio.br

***Abstract* — Interactive storytelling in games is a powerful tool to create immersive and engaging experiences for players. In this context, the adherence to a predefined story arc, coupled with adaptation to individual personality traits, are essential to ensure, at the same time, thematic consistency and player involvement in story-driven games. One promising way to meet such requirements is to treat plot composition as an interactive plan-generation problem, and develop a method whereby branching quests can be adequately handled and adapted in real-time. A key feature of the method is the ability to, after evaluating the effects of player decisions, perform the adaptations needed to keep the current story arc in close approximation to the predefined story arc. The underlying player preference model uses a set of artificial neural networks trained, on the basis of the players' responses to a brief Big Five personality test, to classify their preferences for specific quest decisions. This paper presents our quest adaptation method and summarizes the results we obtained through the application of our method in a fully implemented game prototype.**

Keywords—Quest Adaptation, Automated Planning, Dramatic Structures, Branching Narratives, Interactive Storytelling

I. INTRODUCTION

The design of interactive narratives for games is a challenging task, wherein the adoption of a well-configured story arc is a crucial step to ensure that the players will be led through dramatically engaging sequences of events. In the past, even the compatibility of narrative with the effects of interaction has been questioned due to the lack of control allegedly suffered by authors over narrative structures in interactive media [1][2]. Today, however, interactive storytelling in videogames is a very common practice and many recent games include quests with branching storylines, such as Mass Effect 2 (BioWare, 2010), The Witcher 3: Wild Hunt (CD Projekt RED, 2015), and Cyberpunk 2077 (2020).

When designing interactive narratives for games, one major problem is then how to achieve branching narratives in which all storylines follow a dramatic structure consistent with a predefined story arc. The problem involves the mutual dependence of the players' actions, affected in complex ways by the players' possible narrative choices.

Although story arcs were originally proposed as a way to represent the structure of a dramatic *linear* work, such as a play or film, the new entertainment media has been applying classical dramatic structures for *non-linear* interactive media, such as videogames [3][4]. Considering that non-linearity can be achieved by simply allowing players to customize their linear experiences during a playthrough, traditional narrative structures can still be applied to non-linear branching quests. In this way, one could manually plan all possible storylines of a branching quest to follow an intended narrative structure, but that still does not account for the freedom that players usually have while performing quests. Most games allow players to explore the world (where they can meet new characters, fight enemies, find new items, etc.) while performing the tasks of a quest, and these detours can directly affect the tension of the narrative that is experienced by the player, thus breaking the story arc that was manually drawn for the quest.

In this context, artificial intelligence methods can provide ways to dynamically generate or adapt storylines according to a user-specified story arc. In previous works, we developed prototypes to *generate* quests for games employing automated planning [5], player modeling [7][8]. With these techniques, we were able to obtain an encouraging level of variety in plot composition with high flexibility to support the richness of the mutable interactive virtual worlds of games. We now propose to invest on a strategy that is based on the *adaptation* of already existing branching quests. Instead of generating quests from scratch, an adaptation strategy starts with a sound and coherent quest and modifies its events according to author-specified requirements and/or player preferences.

In this paper, we propose a new quest adaptation method to apply a dramatic structure to branching quests in real-time. By relying on the dynamic structure of quests, which are specified as planning problems, our method can adapt the plot and introduce new events into quests that will lead the player to situations that sufficiently increase or decrease the dramatic tension of the narrative so as to achieve the approximation of the current story arc to the expected story arc.

The paper is organized as follows. Section II presents related work. Section III introduces our quest adaptation method. Section IV describes its application by way of a fully implemented game prototype and presents the results of an evaluation study. Section V offers concluding remarks.

II. RELATED WORK

There are several works on quest generation in the literature, such as the framework presented by Sullivan et al. [9], which uses a rule-based system to dynamically generate the structure of quests according to a library of existing quests. Another recent framework for quest generation was proposed by Ammanabrolu et al. [10], who present a method to generate cooking quests for text-adventure games using Markov chains and a neural language model to make the recipes. A similar approach is explored by Chongmesuk and Kotrajaras [11], but focusing on the analysis of the alternative paths that can be generated for a given type of quest. A more dynamic solution is presented by Lima et al. [5], who propose a method to

generate quests based on hierarchical task decomposition and planning under non-determinism. In a recent work, Lima et al. [7] also proposed a quest generation method based on genetic algorithms and automated planning.

Although both quest generation and adaptation tasks can produce new plots, most of the adaptation methods rely on existing quest plots and use player modeling techniques to infer player preferences. As this paper concerns quest adaptation, we shall focus our analyses of related work on previous works that are centered on this subject. In this context, Li and Riedl [12][13] present an offline refinement search algorithm based on partial order planning that iteratively modifies a pre-existing storyline (deleting and inserting quests and events) until it matches the preferences of a target player. The method we propose here is also based on planning, but instead of adapting the storyline by way of an offline procedure, it performs all the adaptations in real-time. Li and Riedl's technique is limited to previously known player's preferences, and does not account for preference changes and player actions occurring during the game. In contrast, our method relies on the dramatic structure of the narrative, which is directly affected by in-game player actions.

A framework to adapt game narratives according to player's emotions is presented by Freilão [14]. However, all the adaptations are predetermined by the author and no narrative generation method is employed in the process. A more complex emotion-based for narrative adaptation method is explored by Hernandez et al. [15][16], who present a system called PACE, which lets the author specify the emotions that players should experience during a narrative-based game. Their system selects narrative segments to be presented to the player so as to provoke emotional reactions that are close to the emotions the author intended. Despite allowing authors to specify the emotions that players will experience in narrative-based games, PACE does not provide the level of freedom offered by games that are more open to player interactions, such as role-playing games (RPGs). To meet this requirement, our method adapts the plot of quests according to the dramatic structure of the narrative, which is represented as a story arc that is updated in response to all in-game player actions.

A similar approach is explored by Zook et al. [17], who present a skill-based mission generation system that uses player modeling and genetic algorithms to create the game missions. Their system uses the player model to predict a player's performance in response to the mission's challenges and then compares the predicted performance to the author's desired performance. The concept of a target player performance curve is similar to how we define a desired story arc in the present work, but these two concepts have different meanings: while player performance is related to the difficulty of the game, the story arc is related to the dramatic tension of the narrative. In addition, the system of Zook et al. does not perform plot adaptations while the player is progressing through a mission. In contrast, our method adapts the plot of quests in real-time to compensate for player actions that may deviate the current story arc from the author's intended arc.

III. Adaptive Branching Quests

A. The Structure of Branching Quests

In games, a quest represents a journey in which the protagonist (controlled by the player) performs tasks in order to overcome challenges and achieve meaningful goals [18]. Accordingly, in this paper, we define a quest as a set of tasks (e.g. killing enemies, escorting and saving characters, collecting and delivering items) which the player must accomplish. The success in achieving these tasks or the different decisions made by players while performing those tasks will lead them to experience a unique and customized plot. In the present article, we call this unique plot a *storyline*.

In this work, we use the same definition of quests used in [8], where quests are modeled as a tree structure named *branching quest* (Ψ) (Fig. 1), that is:

- The root node represents the initial state of the quest (S_0);
- Internal nodes define intermediate goal states (G_i) and intermediate states (S_i);
- Leaf nodes define final goals and final states for the quest;
- A branch (E_i) is composed of a pair of nodes (S_j, G_i) and an edge ($S_j \rightarrow G_i$), $j \prec i$, where the edge comprises a sequence of events to achieve the intermediate or final goal G_i (and state S_i) from the initial state or the intermediate state that precedes S_i (i.e., S_j). For example, E_4 is composed of (S_1, G_4) and ($S_1 \rightarrow G_4$).

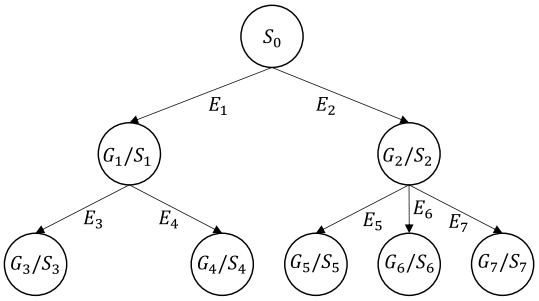

Fig. 1. Tree structure of a branching quest.

Each branch of the quest tree is encoded as a planning problem:

$$E_i = (F, S_j, G_i, O), j \prec i,$$

where F is a set of atomic formulas (or atoms, for short), O is a set of planning operators, $S_j \subseteq F$ is the initial state of E_i, and $G_i \subseteq F$ is the goal state in the form of a ground literal. A literal is an atom f or the negation of an atom ($\neg f$). In this definition, a planning operator $o \in O$ is denoted by:

$$o = \big(name(o), precond(o), effect(o), tension(o)\big)$$

where $name(o)$ is the name of the operator in the form of an atom $op(x_1, x_2, \ldots, x_k)$, $precond(o)$ and $effect(o)$ are sets of literals that define the preconditions and the effects of o. A detailed explanation of these basic terms used in automated planning can be found in our previous works [6][7]. In contrast with the definition used in [8], we extend planning operators with the component $tension(o)$, which establishes how o affects the overall tension of the quest's narrative, which can be increased (+), decreased (–), or maintained (=). As an example, the first branch E_1 of quest Ψ_1 is the planning problem $E_i = (F, S_0, G_1, O)$, where $O = \{o_1, o_2, o_3\}$, and

$name(o_1) =$ `go(CH,PL1,PL2)`,

$name(o_2) =$ `see-starving(CH1,CH2,PL)`,

$name(o_3)$ = `request-another(CH1,CH2,CH3,IT,PL).`

The variables in these operators are characters (CH), places (PL) and items (IT). As a complete example of operator, we present `request-another`:

```
o3:
  name: request-another(CH1, CH2, CH3, IT, PL)
  precond: character(CH1), character(CH2),
           character(CH3),item(IT),place(PL),
           at(CH1,PL),at(CH2, PL), alive(CH1),
           alive(CH2), healthy(CH2),
           hero(CH2), know-need(CH1, CH3, IT)
  effect: know-request(CH2, CH1, IT)
  tension: +
```

An example of goal is:

```
G1: know-request(james, elizabeth, food1),
    at(james, store)
```

By solving the planning problem of a branch using a planning algorithm, a linear sequence of plot events is generated from the initial state S_j. In our implementation, we used the HSP2 heuristic search planner of Bonet and Geffner [19], which is compatible with our STRIPS-based formalism.

The plot of each branch is a sequence of events $\Psi_j E_k = \{e_1, e_2, \ldots, e_n\}$. Each event e_i is a pair (ev_i, st_i), where ev_i is a ground literal that describes the event, and st_i is a set of ground literals that describe the world state that holds after the event e_i. In the above example, the generated plot for $\Psi_1 E_1$ is $\{e_1, e_2, e_3\}$, where events are highlighted in **bold** and new ground literals added to the state by the current event operator are highlighted with underline:[1]

$\Psi_1 E_1$:
e_1:
ev_1=**see-starving(elizabeth, karen, shelter),**
st_1={healthy(james), healthy (elizabeth), open(shelter), open(store), at(james, shelter), at(elizabeth, shelter), at(karen, shelter), starving(karen), know-need(elizabeth, karen, food1)}
e_2:
ev_2=**request-another(elizabeth, james, food1, shelter),**
st_2={healthy(james), healthy (elizabeth), open(shelter), open(store), at(james, shelter), at(elizabeth, shelter), at(karen, shelter), starving(karen), know-need(elizabeth, karen, food1), know-request(james, elizabeth, food1)}
e_3:
ev_3=**go(james, shelter, store),**
st_3={healthy(james), healthy (elizabeth), open(shelter), open(store), at(elizabeth, shelter), at(karen, shelter), starving(karen), know-need(elizabeth, karen, food1), know-request(james, elizabeth, food1), at(james, store)}

This simple plot starts with the character Elizabeth watching her daughter Karen starving. Then, Elizabeth requests food to James (the character controlled by the player), who goes to the store looking for the requested item.

When the planner solves the planning problem of a branch, the resulting plan defines the branch's storyline. In addition, the final state of this plan is used to establish the intermediate state of the branch's child node. This child node can then be used as the initial state to continue the story towards the goals of successor branches.

Fig. 2 shows an example of branching quest (Ψ_1) designed using our formalism. Considering that the plot of $\Psi_1 E_1$ was shown in the previous example, the plot for the remaining branches E_i of Ψ_1 are (for simplicity, we only present the literals ev_i that describe the events of each branch):

$\Psi_1 E_2$ = `ask(james, michael, food1, store).`

$\Psi_1 E_3$ = `steal(james, michael, food1, store), go(james, store, shelter), deliver(james, elizabeth, food1, shelter), feed(elizabeth, karen, food1, shelter).`

$\Psi_1 E_4$ = `request-payment(michael, james, store), pay(james, michael, store), give(michael, james, food1, store), go(james, store, shelter), deliver(james, elizabeth, food1, shelter), feed(elizabeth, karen, food1, shelter).`

$\Psi_1 E_5$ = `request-kill(michael, james, villagemonster1, store), go(james, store, village), kill(james, villagemonster1, oldgun, village), go(james, village, store), report-kill(james, michael, villagemonster1, store), give(michael, james, food1, store), go(james, store, shelter), deliver(james, elizabeth, food1, shelter), feed(elizabeth, karen, food1, shelter).`

$\Psi_1 E_6$ = `request(michael, james, hospitalstoragekey, store), go(james, store, hospital), steal(james, steven, hospitalstoragekey, hospital), go(james, hospital, store), give(james, michael, hospitalstoragekey, store), give(michael, james, food1, store), go(james, store, shelter), deliver(james, elizabeth, food1, shelter), feed(elizabeth, karen, food1, shelter).`

The tree structure of a branching quest can be manually created by a game designer or automatically generated by procedural content generation algorithms. In the former case, a human game designer is responsible for defining the logical description of the initial state for the quest (the game world state in which the quest can start), as well as for establishing intermediate and final goals for the different branches of the quest. On the other hand, when quests are created by procedural content generation techniques, the entire structure of the quest tree is generated by an algorithm, including the definition of the initial state and goals (as described in a previous work of our group [8]). In both cases, the planning problem of the branches is solved by a planner in real-time to

[1] Static literals (i.e., literals that never change in this example), such as `character(CH)`, `alive(CH)`, and `path(PL1, PL2)`, were omitted from the state descriptions.

generate the final storyline as the player progresses through the quest during the game. This dynamic structure provides an opportunity for the development of new algorithms that can adapt the plot of quests in real time according to variables that can only be instantiated during the game, such as individual player preferences, behaviors, or the dramatic structure of the narrative, which can be directly affected by player actions.

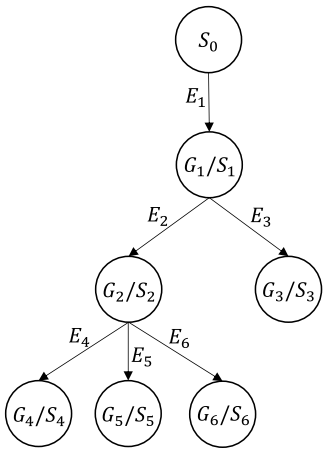

Fig. 2. Tree structure of branching quest Ψ_1.

B. *The Representation of Story Arcs*

In classical non-interactive media (e.g. literature, theatre, novels), the dramatic structure of a work is implied by the form of its narrative and the dramatic impact of its events. In the context of our work, the concept of story arcs stands out as a normative way to represent narrative structures. A popular story arc is the three-act structure (Fig. 3a), which is commonly used by the film industry and is divided into Setup (1/4 of the story time), Confrontation (2/4 of the story time), and Resolution (1/4 or less of the story time). We can use this story arc or any other tension function (see [20] for an overview on story arcs). In the present work, time is discrete. Moreover, we linearize the tension function by parts and assign values to the plot points using unit increments (we propose to call this function a *piecewise linear story arc*). This is a simple way of having a flexible, standard story arc to be used as the *desired arc* for the whole story. Fig. 3b is the piecewise linear three-act story arc used in our experiments.

Although some modern literary scholars tend to be hostile to norms of structure [21], the new entertainment media, such as videogames, have special needs and challenges as they engage players in short narrative episodes (quests and side-quests) with strong focus on the interactive aspects of the experience. In this case, normative notions of dramatic structure are very helpful to increase player engagement.

We represent the piecewise linear story arc of a plot d as a sequence of symbols $d_{arc}^{sym} = \{s_1, s_2, \dots, s_n\}$, where each s_i can be: "+" to indicate rise; or "-" to indicate fall; or "=" to indicate that the tension level is maintained. The number of symbols in the sequence represents the discretized time axis. For example, the three-act story arc d illustrated in Fig. 3b can be expressed as: $d_{arc}^{sym} = \{+, +, +, -\}$.

The symbolic representation of a story arc can also be converted into a numeric representation $d_{arc}^{num} = \{v_1, v_2, \dots, v_n\}$, where each v_i of d_{arc}^{sym} is a number indicating the current tension value in the vertical axis of the story arc. We propose to start the function with zero, and add 1, subtract 1 or do nothing if the symbol is "+", "–", or "=" respectively. For example, the symbolic story arc $d_{arc}^{sym} = \{+, +, +, -\}$ yields to $d_{arc}^{num} = \{1, 2, 3, 2\}$.

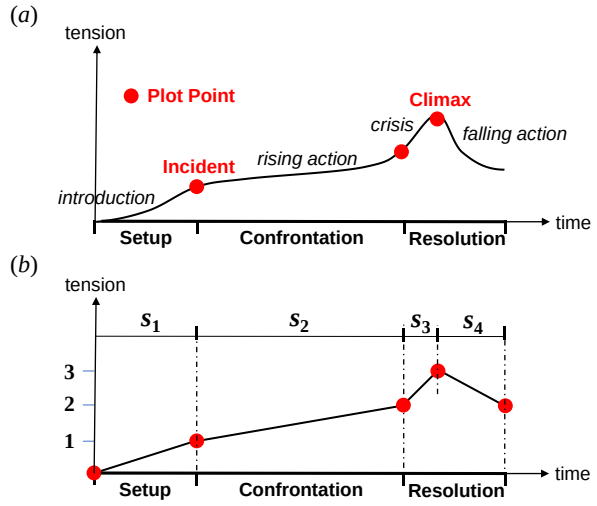

Fig. 3. Examples of story arcs: (*a*) the well-known three-act story arc; and (*b*) a piecewise linear three-act story arc.

The storyline generated for a branching quest can also be expressed in this notation. For example, the story arc for a storyline created by a player P_1 traversing the tree structure of branching quest Ψ_1 (as in Section III – A), following branches $E_1 + E_2 + E_6$, can be rendered in symbolic notation as:[2]

$$P_1{\Psi_1}_{arc}^{sym} = \{+, +, =, +, +, =, +, =, -, -, =, -, -\},$$

and converted to its numeric representation:

$$P_1{\Psi_1}_{arc}^{num} = \{1, 2, 2, 3, 4, 4, 5, 5, 4, 3, 3, 2, 1\},$$

Considering that story arcs can have different time and tension scales, a normalization procedure to scale them to standard intervals is needed for further comparisons. In our experiments, time is scaled to the interval [1, 10], and tension to the interval [0, 1]. The normalization formula to calculate the scaled story arc of a plot p is the same used in [8], reproduced below to help understanding the current extension:

$$\forall_j \in \{1, \dots, 10\}\ p_{arc_j}^{scaled} = \frac{p_{arc}^{num}\left\lfloor \frac{j-1}{10}\left(\overline{p_{arc}^{num}}-1\right)\right\rfloor+1 - \min(p_{arc}^{num})}{\max(p_{arc}^{num}) - \min(p_{arc}^{num})} \quad (1)$$

where $\overline{p_{arc}^{num}}$ denotes the length of p_{arc}^{num}, and $\min(p)$ and $\max(p)$ are functions that return the minimum and maximum tension values of p. For example, the three-act story arc $d_{arc}^{num} = \{1, 2, 3, 2\}$ and the $P_1{\Psi_1}_{arc}^{num}$ above are respectively scaled to:

$$d_{arc}^{scaled} = \{0.3, 0.3, 0.6, 0.6, 0.6, 1.0, 1.0, 0.6, 0.6, 0.6\}$$

$$P_1{\Psi_1}_{arc}^{scaled} = \{0.2, 0.4, 0.6, 0.8, 0.8, 1.0, 0.8, 0.6, 0.4, 0.2\}$$

With two story arcs scaled to the same intervals, we can calculate their differences. The direct difference between two points at index i in two scaled story arcs (x and y) is given by:

[2] We used the following tension effects of the operators: `go: =, see-starving: +, request-another: +, ask:+, steal:+, deliver:-, feed:-, request-payment:+, pay:-, give:-, request-kill:+, kill:-, report-kill:-, request:+`.

$$diff(x_i, y_i) = (x_i - y_i)^2 \quad (2)$$

And the difference between two story arcs (p and d) is:

$$arcdiff(p, d) = \frac{1}{n}\sum_{i=1}^{n} diff\left(p_{arc_i}^{scaled}, d_{arc_i}^{scaled}\right) \quad (3)$$

where n is the maximum value used in the scaled time interval of the story arcs (in our experiments, $n = 0$).

Although the mathematical representation of story arcs allows direct comparisons between two story arcs, applying a dramatic structure to interactive and non-linear narratives is a challenging task, giving the fact that traditional narrative structures are inherently linear and are not prepared to handle the freedom that videogames give to players.

The proposed method to apply a dramatic structure to a branching quest involves the adaptation of the quest's plot in real time to approximate the current story arc to an expected story arc. Such approximation requires the full estimation of the current story arc, which must take into account past and future player decisions. Since different storylines can occur during a branching quest, a method to identify the player's preferences for future narrative events is needed to allow the system to predict the most likely storyline that will occur.

C. *Predicting Player's Decisions in Branching Quests*

In order to identify the path that a player will follow in a branching quest, we adopted the method proposed by Lima et al. [22][23], which was originally designed to identify the users' preferences for narrative events using machine learning and the Big Five personality model in an interactive storytelling system. In the present work, we extended and applied the model proposed by Lima et al. [22] in a game context. To assess the personality of players, we directly integrated the BFI-10 [24] questionnaire into our game. So, before starting the game, players must answer the 10 questions of the BFI-10, which will measure their personalities. Although less invasive solutions to integrate the BFI-10 questionnaire into games exist, such as the use of story-related interactive scenes [6], we opted for this simpler solution, since, although assessing players' personalities is part of this work, it is not its main focus.

For the preference model, we used a set of artificial neural networks trained to classify player preferences for specific quest decisions. As illustrated in Fig. 4, each artificial neural network is trained to identify the predilections of players for the possible choices of a branching node in the tree structure of a quest. The neural networks use a single hidden layer and are trained by a standard back-propagation learning algorithm using a sigmoidal activation function. The input for all neural networks is defined by the five scores of the Big Five factors (Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism), which are obtained through the BFI-10 questionnaire. Their output is defined by the possible choices available for their respective branching nodes.

The dataset used to train the artificial neural networks of our preference model was collected from game sessions that occurred as part of a user evaluation test conducted for a previous work on the procedural generation of branching quests [8]. A total of 38 players played a prototype game with the branching quests used for the present work. During the game sessions, players had their personalities assessed through the BFI-10 questionnaire and their decisions on each branching node were automatically registered by the game. The artificial neural networks of the preference model were then trained and evaluated according to a 10-fold cross-validation strategy (using the methodology described by Lima et al. [22]). The results indicate that our preference model has an average accuracy of 87.2%.

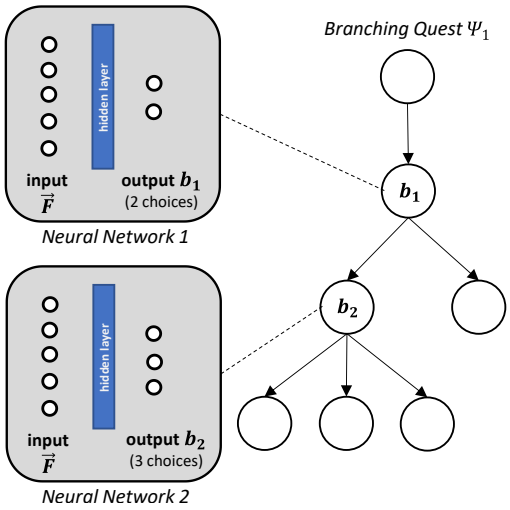

Fig. 4. Structure of neural networks used in the preferece model: $\vec{F}$ is the player personality vector (five values representing the scores of the Big Five factors), and b_i represents the number of choices in the i-th branching point of the quest tree.

The preference model allows the system to predict future player's decisions in branching nodes at any time during the game. With this information, the system can estimate a possible final plot for the branching quest.

D. *The Dynamic Story Arcs of Interactive Branching Quests*

The current story arc for a branching quest is automatically calculated as the player progresses through the events of the quest in real time. By combining past events and player actions with the events that are anticipated for the player (predicted by preference model), the current story arc can be dynamically estimated in response to the actions of the player.

Given that a quest plot is an indexed sequence of events $\Psi_j \mathrm{E}_k = \{e_1, e_2, \ldots, e_n\}$, the player's progress can be determined by the current quest event e_π, where π is the index of the current event. The quest starts with the event e_1 ($\pi = 1$) and ends when the player completes e_n ($\pi = n$). As both symbolic and numeric representations of story arcs are created according to a quest plot, the index π can also be used to mark the progress of the player in the story arc of the quest.

The current story arc is updated in response to player actions (e.g., encountering and fighting enemies, finding and using items, interacting with non-player characters) and plot events from other active quests. In both cases, the symbolic representation of the story arc is updated by adding the tension symbols of the new actions/events at index π according to the tension symbol associated with the operator related to the action/event, which is defined in the elements of the planning problem (see Section III – A).

For example, considering a player P_1 that has just started quest Ψ_1 (presented in Section III – A) and assuming that the preference model indicates that this player will follow branches $E_1 + E_2 + E_5$, the current story arc for P_1 and Ψ_1 can be expressed in the symbolic notation as:

$$P_1\Psi_1{}_{arc}^{sym} = \{\underline{+}, +, =, +, +, =, +, =, -, -, =, -, -\},$$

where the symbol with underbar ($\underline{+}$) indicates the current position of the player in the story arc according to index π.

Assuming that, after starting quest Ψ_1, the player performs the first two events as expected in the quest plot (watches `karen` starve and receives the request of `food` from `elizabeth` in the `shelter`). However, when going to the store (the 3rd event of the quest), the player meets an enemy. The enemy encounter is not part of the quest plot, but it will affect the narrative tension. Therefore, the tension produced by the encounter event is inserted in the story arc at index π:

$$P_1\Psi_1{}_{arc}^{sym} = \{+, \overline{+}, \underline{+}, =, +, +, =, +, =, -, -, =, -, -\},$$

where the overbar ($\overline{+}$) indicates the inserted symbol.

The narrative tension of the quest can also be affected by plot events from other active quests. For example, after killing the `villagemonster1` in the 7th event of the quest, the player may decide to interact with some character needing help in the village to start a new side-quest Ψ_2 (e.g. this character requests the player to find a syringe). Since the plot of Ψ_2 has:

```
request-item(david, james, syringe, village),
go(james, village, hospital), get(james,
syringe, hospital), go(james, hospital,
village),deliver(james,david,syringe,village),
```

and assuming that the player P_1 completes all the events of Ψ_2 before proceeding to the next objective of Ψ_1, the story arc of $P_1\Psi_1$ will be updated and the tension variation values produced by the events of Ψ_2 are inserted at index π:

$$P_1\Psi_1{}_{arc}^{sym} = \{+, +, +, =, +, +, \overline{+, =, +, =, -}, \underline{=}, +, =, -, -, =, -, -\},$$

By calculating and updating the story arc of a branching quest according to player actions, the system can estimate the current story arc of a quest at any time during the game.

E. Adapting the Story Arcs of Branching Quests

The proposed method to adapt the plot of a branching quest to approximate the current story arc to an expected story arc relies on the dynamic structure of the planning problems used to define the quest's branches, which are solved by a planner in real time to generate the final storyline for the quest. By modifying the world state, current goals, and performing replanning procedures, the system can adapt the plot and introduce new events into the quest, leading the player to situations that sufficiently increase or decrease the dramatic tension of the narrative to achieve the approximation of the current story arc to the expected story arc.

The plot adaptions that can be performed by the system are defined in the *adaptation library*, which is manually constructed by a human author according to the type of the game's possible interactions and events. The *adaptation library* is a set $L = \{m_1, m_2, \ldots, m_n\}$, where each member of L is a 5-tuple $m_i = (\gamma_i, \alpha_i, \beta_i, \delta_i, \theta_i)$. The elements of m_i are:

- γ_i defines the **modification effect** of m_i, which can *increase* or *decrease* the dramatic tension of the narrative;
- α_i is a set of literals that establish **event preconditions** (i.e., the narrative events that enable m_i to occur). For example, $\alpha_i =$ {`go(CH, PL1, PL2)`} defines that m_i can only be applied if the current narrative event is a `go` event. In this case, since `CH`, `PL1`, and `PL2` are variable terms, any ground terms are accepted for the narrative event. But if $\alpha_i =$ {`deliver(CH1, elizabeth, IT, PL)`, `deliver(CH1, david, IT, PL)`}, then m_i can only occur if the current event is a `deliver` event where an item `IT` (any item) is being delivered by a character (any character) to `elizabeth` or `david` (ground terms). When $\alpha_i = \emptyset$, there are no event preconditions and m_i can be applied in sequence to any event of the quest;
- β_i is a set of literals defining **state preconditions** (i.e., literals that must hold in the current world state for m_i to be applied). For example, $\beta_i =$ { `know-request(james, CH2, IT)` } establishes that m_i can only occur if `james` knows that he has been requested by a character `CH2` to find and deliver an item `IT`. Since `know-request` is added to the world state as an effect of the `request` operator and removed from the world state as an effect of the `deliver` operator, m_i can occur at any moment while `james` is searching or delivering `IT`. Function symbols can also be used to represent terms. For example, $\beta_i =$ { `at(enemy(EN), player-location(PL))` } defines that m_i can only occur if there is an enemy `EN` (any enemy) at the current player location `PL`. When $\beta_i = \emptyset$, there are no state preconditions;
- δ_i is a set of literals that define **state modifications** (i.e., literals to be added or removed from the current world state where m_i is being applied). For example, considering $\alpha_i =$ {`go(CH, PL1, PL2)`}, the state modifications $\delta_i =$ { `¬open(PL2)`, `at(key(PL2, KE), PL1)`} establish that the place `PL2` will not be open (the negation symbol ¬ represents the deletion of the literal from the current world state) and a key `KE` for `PL2` will be added to the location `PL1`. The values of the variable terms of δ_i are established according to the terms of α_i, which are defined according to the ground terms of the `go` event where m_i is being applied;
- θ_i is a set of literals that define **goal modifications** (i.e., literals to be added or removed from a goal state that is created using as basis the world state holding after the completion of the current event where m_i is being applied). For example, $\theta_i =$ { `open(PL2)` } defines `open(PL2)` as an extra goal to be accomplished by the player before he resumes the original events of the quest. As will be explained below, these goals are used by the planner to generate and introduce new events into the quest plot.

In our implementation, six plot adaptions were tested. Three of them were designed to increase the dramatic tension of the narrative: (1) spawn an enemy at the current player location (forcing the player to fight and kill the enemy); (2) block the player passage (forcing him to find a key or another item to open the passage); and (3) make the player lose or break an item that is being delivered (inducing the player to find another item). The other three plot adaptions were

designed to decrease the tension: (1) spawn an item that the player needs at the current player location (making the player find the item easily); (2) spawn an assistant character at the current player location to kill an enemy (helping the player to eliminate the threat); and (3) spawn an assistant character at the current player location to give the player an item that the player needs (avoiding the item search process).

For example, the plot adaptation to increase the narrative tension by blocking the player passage is represented in the adaptation library as:

$m_1 = \{\gamma_1 = increase, \alpha_1 = \{$`go(CH, PL1, PL2)`$\}, \beta_1 = \emptyset,\ \delta_1 = \{\neg$`open(PL2), at(key(PL2, KE), PL1)`$\},$ $\theta_1 = \{$`open(PL2), know-was-closed(CH, PL1)`$\}\}$.

An example of plot adaptation to decrease the narrative tension involves spawning an item that the player needs at the current player location, shown in the adaptation library as:

$m_2 = \{\gamma_2 = decrease, \alpha_2 = \emptyset,\ \beta_2 = \{$`know-request(james, CH2, IT)`$\},\ \delta_2 = \{$`at(IT, player-location(PL))`$\}, \theta_2 = \{$`has(james, IT)`$\}\}$.

Our quest adaptation algorithm is associated with a quest instance, and maintains the symbolic representation of the current story arc for the quest, which is initially estimated by the preference model (as described in Section III – C). Every time the player performs a relevant action (i.e., an action that can affect the narrative tension), the algorithm compares the current tension values of the current story arc and the desired story arc to decide whether a plot adaptation is necessary or not. A threshold value Ω defines the maximum acceptable error for the difference between the tension values of the story arcs. In our experiments, we let $\Omega = 0.07$.

Once given as input the current story arc ${P_j\Psi_k}_{arc}^{sym}$, a desired story arc d_{arc}^{sym} (both in the symbolic notation), and the index of the current event π, the process to adapt a branching quest in real time comprises the following steps:

1. Calculate the scaled story arcs ${P_j\Psi_k}_{arc}^{scaled}$ and d_{arc}^{scaled} using Equations (1) and (2) (defined in Section III – B), according to ${P_j\Psi_k}_{arc}^{sym}$ and d_{arc}^{sym};
2. Calculate the difference between the tension values of ${d_{arc}^{scaled}}_{\pi}$ and ${P_j\Psi_k}_{arc\ \pi}^{scaled}$ using Equation (3);
3. If $diff\left({d_{arc}^{scaled}}_{\pi}, {P_j\Psi_k}_{arc\ \pi}^{scaled}\right) < \Omega$:
    a. No plot adaptations are necessary.
4. Otherwise:
    a. Identify the modification type (λ) required for the adaptation:
        i. If ${d_{arc}^{scaled}}_{\pi} > {P_j\Psi_k}_{arc\ \pi}^{scaled}$:
            1. $\lambda = increase$;
        ii. Otherwise:
            1. $\lambda = decrease$;
    b. Get from the adaptation library L all plot modifications of type $\gamma_i = \lambda$ where α_i and β_i hold in the current event and world state;
    c. Simulate the application of all accepted plot modifications in the current quest, which will produce a set of plot variants $V = \{var_1, var_2, \ldots, var_n\}$.
    d. Calculate the scaled story arcs ${var_i}_{arc}^{scaled}$ for all simulated plot variants $var_i \in V$.
    e. Compare the story arcs ${var_i}_{arc}^{scaled}$ and d_{arc}^{scaled} using Equation (4), and then select the plot var_i whose story arc ${var_i}_{arc}^{scaled}$ produces the smallest error in comparison with d_{arc}^{scaled};
    f. Update the plot of the current quest Ψ_k according to the selected var_i.

When applying a plot modification m_i to a quest, the state modifications $\delta_i \in m_i$ are directly used to modify the current world state of the quest. In addition, the goal modifications $\theta_i \in m_i$ are used to establish new intermediate goals for the current event e_i. These elements are used to define a new planning problem, where the current world state (modified according to $\delta_i \in m_i$) is used to establish the initial state (S_0), and the ground literals of the state $st_i \in e_i$ (the state that hold after e_i), complemented with the goals of θ_i, are used to establish the goal state (G_i). The planning problem is then solved by a planner, which generates a new sequence of events to be added to the quest. In order to avoid inconsistencies caused by these new events and state modifications, a replanning procedure is performed in all planning problems of future branches of the quest.

For example, let us consider the following situation:

- A player P_2 is at the 3rd event of quest Ψ_1 ($\pi = 3$);
- The preference model indicates that P_2 will follow branches $E_1 + E_2 + E_4$ (described in Section III – A);
- The three-act story arc is the desired story arc for quest Ψ_1 ($d_{arc}^{sym} = \{+, +, +, -\}$).
- The current story arc is: ${P_2\Psi_1}_{arc}^{sym} = \{+, +, \underline{=}, +, +, -, -, =, -, -\}$.

By converting d_{arc}^{sym} and ${P_2\Psi_1}_{arc}^{sym}$ to their numeric representations, and then scaling them to the same time intervals (time is scaled to the interval [1, 10] and tension is scaled to the interval [0, 1]), the story arcs can be compared (a visual comparison is shown in Fig. 5):

$$d_{arc}^{scaled} = \{0.33, 0.33, \underline{0.66}, 0.66, 0.66, 1.00, 1.00, 0.66, 0.66, 0.66\},$$

$${P_2\Psi_1}_{arc}^{scaled} = \{0.25, 0.50, \underline{0.50}, 0.75, 1.00, 0.75, 0.50, 0.50, 0.25, 0.00\}.$$

With both story arcs scaled to the same intervals, the difference between the tension values of ${d_{arc}^{scaled}}_{\pi}$ and ${P_2\Psi_1}_{arc\ \pi}^{scaled}$ can be calculated:

$$diff\left({d_{arc}^{scaled}}_{\pi}, {P_2\Psi_1}_{arc\ \pi}^{scaled}\right) = (0.66 - 0.50)^2 = 0.0277$$

Considering that $diff\left({d_{arc}^{scaled}}_{\pi}, {P_2\Psi_1}_{arc\ \pi}^{scaled}\right)$ is less than Ω, no interferences in the plot are required at this point.

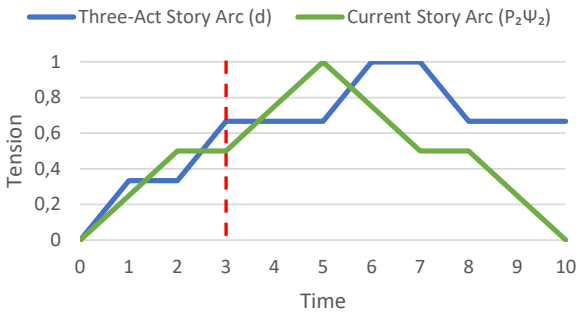

Fig. 5. Visual comparison beteween the desired story arc d and the current story arc for quest $P_2\Psi_1$. The doted red line indicates the current player position in the story arc.

However, the player may decide to explore the world on his way to the store and may end up finding and using a heal item (the item is automatically used when collected), which – according to the tension effect of `heal` operator – reduces the tension of the narrative. The action of collecting/using the item will trigger another run of the quest adaptation procedure to verify the consistency of the current story arc. Considering the `heal` event, the symbolic representation of the current story arc is updated to:

$$P_2\Psi_{1_{arc}}^{sym} = \{+, +, -, \underline{=}, +, +, -, -, =, -, -\},$$

then it is scaled to:

$$P_2\Psi_{1_{arc}}^{scaled} = \{0.33, 0.66, 0.33, \underline{0.33}, 0.66, 1.00, 0.66, 0.33, 0.33, 0.00\},$$

and the difference between the tension values of $d_{arc}^{scaled}{}_{\pi}$ and $d_{arc}^{scaled}{}_{\pi}$ can be calculated as:

$$diff\left(d_{arc}^{scaled}{}_{\pi}, P_2\Psi_{1_{arc}}^{scaled}{}_{\pi}\right) = (0.66 - 0.33)^2 = 0.1111$$

As result, $diff\left(d_{arc}^{scaled}{}_{\pi}, P_2\Psi_{1_{arc}}^{scaled}{}_{\pi}\right)$ will be greater than Ω (0.07), so a plot adaptation procedure is required to approximate the current story arc to the desired story arc (a visual comparison of both story arcs is presented in Fig. 6).

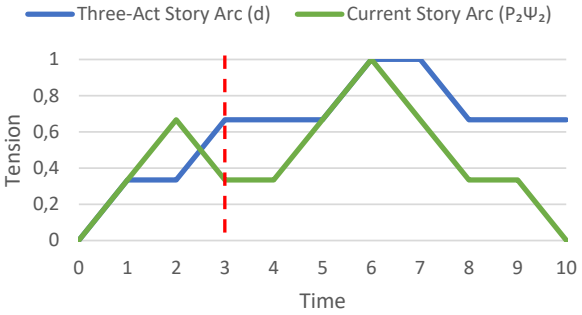

Fig. 6. Visual comparison beteween the desired story arc d and the current story arc for quest $P_2\Psi_1$ after the player finding and using a heal item. The doted red line indicates the current player position in the story arc.

Considering that $d_{arc}^{scaled}{}_{\pi} > P_2\Psi_{1_{arc}}^{scaled}{}_{\pi}$, the dramatic tension of the current story arc must be increased, therefore $\lambda = increase$. By testing the event and state preconditions (α_i and β_i) of the plot adaptions $m_i \in L$ where $\gamma_i = \lambda$, two possible plot adaptions for the current event are identified: (1) block the player passage ($m_1 \in L$); and (2) spawn an enemy at the current player location ($m_3 \in L$).

After identifying the possible plot adaptations, their application in the current state is simulated and evaluated. As previously presented, the state modifications $\delta_1 \in m_1$ comprise: `¬open(PL2), at(key(PL2, KE), PL1)`. Both `PL2` and `PL1` are variable terms instantiated according to the event precondition ($\alpha_1 = \{$`go(CH, PL1, PL2)`$\}$) and the current event where the precondition holds (`go(james, shelter, store)`) . The function symbol `key` is also instantiated by identifying the object that is a key to open `PL2` (`store`), which is called `keystore1`. Therefore, the state modifications are instantiated as ground terms: `¬open(store), at(keystore1, shelter)`. Similarly, the goal modification $\theta_1 \in m_1$ is also instantiated as: `open(store), know-was-closed(james, store)`.

Assuming that the current world state comprises:

```
healthy(james), healthy (elizabeth),
open(shelter), open(store), at(elizabeth,
shelter), at(karen, shelter),starving(karen),
know-need(elizabeth, karen, food1), know-
request(james, elizabeth, food1), at(james,
village).
```

and since that the state that holds after the current event is:

```
healthy(james), healthy (elizabeth),
open(shelter), open(store), at(elizabeth,
shelter), at(karen, shelter),
starving(karen), know-need(elizabeth, karen,
food1), know-request(james, elizabeth,
food1), at(james, store).
```

a new planning problem can be defined, with initial state S_0:

```
healthy(james), healthy (elizabeth),
open(shelter), at(elizabeth, shelter),
at(karen, shelter),starving(karen), know-
need(elizabeth, karen, food1), know-
request(james, elizabeth, food1), at(james,
village), at(keystore1, shelter).
```

and with goal state G_i comprising:

```
healthy(james), healthy (elizabeth),
open(shelter), at(elizabeth, shelter),
at(karen, shelter), starving(karen), know-
need(elizabeth, karen, food1), know-
request(james, elizabeth, food1), at(james,
store), open(store), know-was-closed(james,
store).
```

After solving the planning problem, a new sequence of events is generated, which is then added to the quest plot to define a plot variant. Replanning procedures are performed in all remaining quest branches to guarantee the logical consistency of the plot. The resulting plot variant is (new events were highlighted in bold):

var_1 = ..., **`fail-to-open(james, store, village), go(james, village, shelter), getkey(james, keystore1, shelter), go(james, shelter, store), open(james, store, keystore1, village),`** `go(james, village, store), ask(james, michael, food1, store), request-payment(michael, james, store), pay(james, michael, store), give(michael, james, food1, store), go(james, store, shelter), deliver(james, elizabeth, food1, shelter), feed(elizabeth, karen, food1, shelter).`

A similar process is performed for plot adaptation m_3, which produces another plot variant:

var_2 = ..., **kill(james, monster1, village)**, go(james, village, store), ask(james, michael, food1, store), request-payment(michael, james, store), pay(james, michael, store), give(michael, james, food1, store), go(james, store, shelter), deliver(james, elizabeth, food1, shelter), feed(elizabeth, karen, food1, shelter).

After generating all plot variants, their story arcs are calculated:

$$var_{1}{}_{arc}^{scaled} = \{0.33, 0.33, 0.66, 1.0, 1.0, 0.66, 1.00, 0.66, 0.33, 0.00\}$$

$$var_{2}{}_{arc}^{scaled} = \{0.25, 0.25, 0.50, 0.50, 0.75, 1.00, 0.75, 0.50, 0.25, 0.00\}$$

and then are compared with the desired story arc:

$$arcdiff\left(d_{arc}^{scaled}, var_{1}{}_{arc}^{scaled}\right) = 0.0888$$

$$arcdiff\left(d_{arc}^{scaled}, var_{2}{}_{arc}^{scaled}\right) = 0.0784$$

Both adapted plots improve the story arc of the quest (because the difference between the previous and the desired story arcs was 0.1111). Since $arcdiff\left(d_{arc}^{scaled}, var_{2}{}_{arc}^{scaled}\right) < arcdiff\left(d_{arc}^{scaled}, var_{1}{}_{arc}^{scaled}\right)$, the plot generated for var_2 is used to update the current plot of quest Ψ_1.

IV. Application and Evaluation

The game used to test and evaluate the proposed quest adaptation method is a 2D RPG developed for a previous project [5][6][7][8], in which we incorporated the proposed system to adapt the plot of branching quests in real-time (Fig. 7). The game pertains to a zombie survival genre, telling the story of a family that lives in a world dominated by a zombie plague. The gameplay is designed to be driven by the story quests, but the player is free to explore the game world. While performing a quest or exploring the world, players can collect items, deliver items, interact with non-player characters, kill enemies, open locked doors, fix broken bridges, cure infected characters, and feed those who starve. The game comprises 14 story quests, of which 3 were created by a professional game designer and 11 were generated by a procedural quest generation algorithm (as described in [8]). The game also includes 4 side-quests (quests that are not related to the main story of the game), which were algorithmically generated. More details about the game are presented in [7] and [8].

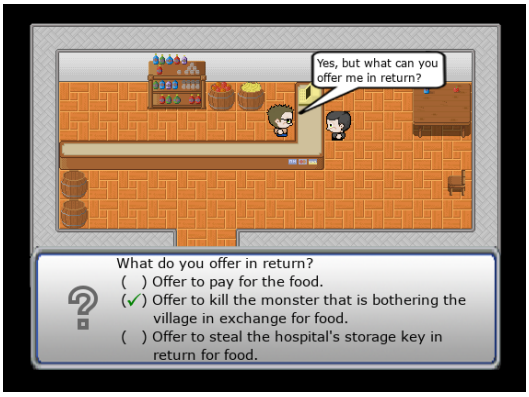

Fig. 7. Scene from the game prototype: the player is faced with the decision that will lead quest Ψ_1 to branches E_4, E_5, or E_6.

To evaluate the proposed quest adaptation method, we conducted a user test to analyze the progress of real players through the dynamic plot of the adaptive quests. A total of 12 volunteers participated in the study (11 bachelor's students and 1 master's student). Ten subjects were male and two female. Ages ranged from 18 to 25 years (mean of 20.3). All of them play video games at least weekly.

For the experiment, we created two versions of our game: (1) *Adaptive Version*, which fully uses the proposed method to adapt the plot of quests using the three-act story arc as the desired story; and (2) *Base Version*, wherein the quest adaptation process is disabled. Both versions were designed to automatically capture and store all player actions, decisions, and the story arcs experienced by players in all quests.

The participants were divided into two groups: 6 of them were randomly selected to play the *Adaptive Version*, and the other 6 participants played the *Base Version*. Before testing the game, all subjects filled a consent form, answered a basic demographic questionnaire, and then were asked to freely play our game. To avoid biased experiences, we did not mention to participants that the game was adapting the plot of quests.

Although we applied our quest adaption method in all quests of the *Adaptive Version* of the game, not all players were experiencing the same quests during a single playthrough of the game. As described in [8], our game adopts a tree structure to establish hierarchical dependencies between story quests, which defines the storyline of the game and the quests made available according to player decisions in previous branching quests. Therefore, we shall focus the analysis for this study in the first quest of the game Ψ_1 (described in Section III), which is played by all players and allows us to compare both versions of the game.

All participants were able to complete the game. On average, each session of the *Base Version* lasted 15.6 minutes (standard deviation 3.1), and each session of the *Adaptive Version* lasted 19.7 minutes (standard deviation 4.3). The average time required by players to complete quest Ψ_1 on the *Base Version* was 4.8 minutes (standard deviation 3.3), and 6.9 minutes on the *Adaptive Version* (standard deviation 3.8).

To analyze the story arcs experienced by players in quest Ψ_1, we compared the differences between the story arcs effectively experienced by players and the desired story arc (using equation (4)). The average difference between the story arcs in the *Base Version* was 0.0936 (standard deviation 0.0272). In the *Adaptive Version*, the average difference was 0.0282 (standard deviation 0.0113), which is more than three times smaller than the difference obtained for the *Base Version*. Fig. 8 shows the best and worst story arcs for the *Adaptive Version* as compared to the desired story arc. The best and worst story arcs for the *Base Version* are in Fig. 9.

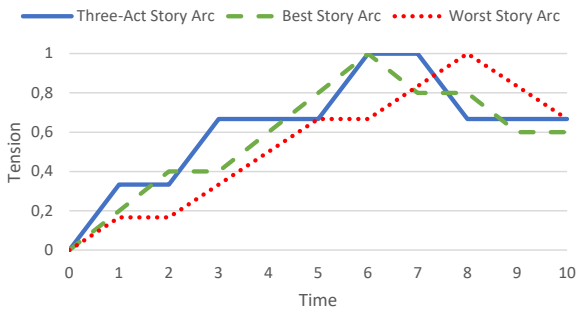

Fig. 8. Comparison of the three-act story arc with the best and worst story arcs experienced by players in quest Ψ_1 of the *Adaptive Version*.

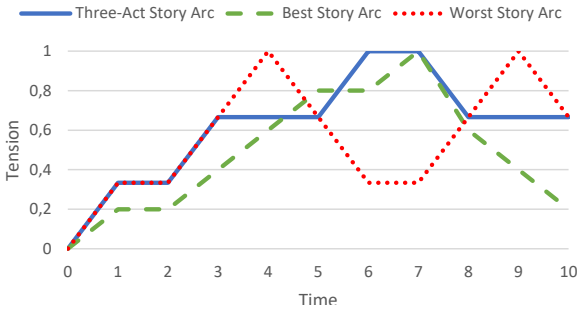

Fig. 9. Comparison of the three-act story arc with the best and worst story arcs experienced by players in quest Ψ_1 of the *Base Version*.

V. Concluding Remarks

The paper presented a novel quest adaptation method combining automated planning, player modeling and dramatic structures. Starting with sound and coherent branching quests designed by professional game designers or by procedural content generation algorithms, the method is capable of adapting the plot of quests in real-time by introducing new events that will lead the player to situations that sufficiently increase or decrease the dramatic tension of the narrative.

The results of our experiment show that the proposed method is indeed capable of approximating the story arcs of quests to an expected story arc, as players progress through the narrative and freely interact with the game world. The time required by players to complete the adapted quests is of course longer than in non-adaptive quests, since the adaptations introduce new events into the plot. This, however, did not prevent the participants to complete the game, and their positive feedback, especially the enthusiasm demonstrated by them when they were told that the game was adapting the quests according to their actions, is a welcome stimulus for the continuation of our work.

Apart from our commitment to validate our method in more rigorous user studies, one promising future work that caught our attention is the automatic measurement of the tension produced by each type of quest event according to past player data. Currently, these effects are manually defined by human authors, who have to judge which events can increase or decrease the dramatic tension of the narrative according to their past experience. In this context, the use of sensors to collect players' biometric data and vital signals in combination with machine learning techniques, can be useful to identify patterns in the emotional reactions of players. These patterns would then be used by our quest adaptation method as a source of information about the dramatic impact of each narrative event, in order to attain a closer conformity between system-generated and predefined story arcs.

Acknowledgments

We would like to thank CNPq (National Council for Scientific and Technological Development) and FINEP (Brazilian Innovation Agency), which belong to the Ministry of Science, Technology, and Innovation, for the financial support.

References

[1] J. Juul, "A clash between game and narrative. A thesis on computer games andinteraction fiction," M.S. Thesis, University of Copenhagen, Copenhagen, Denmark, 1999.

[2] E. Adams, "The Designer's Notebook: Three Problems for Interactive Storytellers," Gamasutra, 1999. [online]. Available at: http://www.gamasutra.com/view/feature/131821/the_designers_notebook_three_.php

[3] B. Ip, "Narrative Structures in Computer and Video Games: Part 1: Context, Definitions, and Initial Findings," Games and Culture, vol. 6 (2), pp. 103-134, 2011, doi: 10.1177/1555412010364982.

[4] B. Ip, "Narrative Structures in Computer and Video Games: Part 2: Emotions, Structures, and Archetypes," Games and Culture, vol. 6 (3), pp. 203-244, 2011, doi: 10.1177/1555412010364984.

[5] E. S. Lima, B. Feijó, and A. L. Furtado, "Hierarchical Generation of Dynamic and Nondeterministic Quests in Games," in Proceedings of the 11th International Conference on Advances in Computer Entertainment Technology, Funchal, Portugal, 2014, Article N. 24.

[6] E. S. Lima, B. Feijó, and A. L. Furtado, "Player Behavior and Personality Modeling for Interactive Storytelling in Games," Entertainment Computing, vol. 28, pp. 32-48, 2018.

[7] E. S. Lima, B. Feijó, and A. L. Furtado, "Procedural Generation of Quests for Games Using Genetic Algorithms and Automated Planning," in Proceedings of the XVIII Brazilian Symposium on Computer Games and Digital Entertainment (SBGames 2019), Rio de Janeiro, Brazil, 2019, pp. 495-504, doi: 0.1109/SBGames.2019.00028

[8] E. S. Lima, B. Feijó, and A. L. Furtado, "Procedural Generation of Branching Quests for Games," Entertainment Computing, 2022.

[9] A. Sullivan, M. Mateas, and N. Wardrip-Fruin, "Rules of engagement: Moving beyond combat-based quests," in Proceedings of the Intelligent Narrative Technologies III Workshop (INT3 '10), 2010, Article No. 11.

[10] P. Ammanabrolu, W. Broniec, A. Mueller, J. Paul, and M. O. Riedl, "Toward Automated Quest Generation in Text-Adventure Games," arXiv:1909.06283 [cs.CL], 2019.

[11] T. Chongmesuk, and V. Kotrajaras, "Multi-Paths Generation for Structural Rule Quests," in Proceedings of the 16th International Joint Conference on Computer Science and Software Engineering (JCSSE), pp. 97-102, 2019, doi: 10.1109/JCSSE.2019.8864168.

[12] B., Li, and M. O., Riedl, "Planning for Individualized Experiences with Quest-Centric Game Adaptation," in Proceedings of the ICAPS 2010 Workshop on Planning in Games, Toronto, Canada, 2010.

[13] B. Li, and M. O. Riedl, "An Offline Planning Approach to Game Plotline Adaptation," in Proceedings of the 6th Conference on Artificial Intelligence for Interactive Digital Entertainment, Palo Alto, California, pp. 45-50, 2010.

[14] M. Freilão, "Affective Narratives for Engagement in Digital Games," M.S. Thesis in Computer Engeneering, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal, 2020.

[15] S. P. Hernandez, V. Bulitko, and M. Spetch, "Keeping the Player on an Emotional Trajectory in Interactive Storytelling," in Proceedings of the 11th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, pp. 65-71, 2015.

[16] S. P. Hernandez, V. Bulitko, and E. Hilaire, "Emotion-based interactive storytelling with Artificial Intelligence," in Proceedings of the Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'14), pp. 146-152, 2014.

[17] A. Zook, S. Lee-Urban, M. R. Drinkwater, and M. O. Riedl, "Skill-based Mission Generation: A Data-driven Temporal Player Modeling Approach," in Proceedings of the The third workshop on Procedural Content Generation in Games (PCG'12), pp. 1-8, 2012.

[18] J. Howard, Quests: Design, Theory, and History in Games and Narratives. Natick, Massachusetts: A K Peters/CRC Press, 2008.

[19] B. Bonet, and H. Geffner, "Planning as Heuristic Search," Artificial Intelligence, vol. 129 (1), pp. 5-33, 2001.

[20] J. Yorke, Into The Woods: How Stories Work and Why We Tell Them. London, UK: Penguin, 2014.

[21] H. Koenitz, A. Di Pastena, D. Jansen, B. de Lint, and A. Moss, "The Myth of 'Universal' Narrative Models," in Rouse R., Koenitz H., Haahr M. (eds) Interactive Storytelling. ICIDS 2018. Lecture Notes in Computer Science, vol. 11318, Springer, 2018.

[22] E. S. Lima, B. Feijó, and A. L. Furtado, "Adaptive Storytelling Based on Personality and Preference Modeling," Entertainment Computing, vol. 34, 100342, 2020, doi: 10.1016/j.entcom.2020.100342.

[23] E. S. Lima, B. Feijó, A. L. Furtado, and V. M. Gottin, "Personality and Preference Modeling for Adaptive Storytelling," in Proceedings of the XVII Brazilian Symposium on Computer Games and Digital Entertainment, Foz do Iguaçu, Brazil, 2018, pp. 538-547.

[24] B. Rammstedt, and O. P. Johnb, "Measuring personality in one minute or less: A 10-item short version of the Big Five Inventory in English and German," Journal of Research in Personality, vol. 41 (1), pp. 203-212, 2007, doi: 10.1016/j.jrp.2006.02.001.