An Optimization Approach to Believable Behavior in Computer Games

Yifeng Zeng[1], Hua Mao[1], Fan Yang[2], and Jian Luo[2]

[1] Department of Computer Science, Aalborg University, Denmark
{yfzeng,huamao}@cs.aau.dk
[2] Department of Automation, Xiamen University, China
{yang,jianluo}@xmu.edu.cn

Abstract. Many artificial intelligence techniques have been developed to construct intelligent non-player characters (NPCs) in computer games. As games are gradually becoming an integral part of our life, they require human-like NPCs that shall exhibit believable behavior in the game-play. In this paper, we present an optimization approach to designing believable behavior models for NPCs. We quantify the notion of believability using a multi-objective function, and subsequently convert the achieving of believable behavior into one function optimization problem. We compute its analytical solutions and demonstrate the performance in a practical game.

1 Introduction

Designing intelligent non-player characters (NPCs) has been a focus of game designers and developers who often resort to sophisticated techniques in the area of artificial intelligence (AI). As expected, the resulting NPCs make smart, nearly optimal, decisions in a complex game world. For example, a tennis NPC may strongly attack human players in the virtual tennis game [16] and you may be impressed by tricky plans of *Frederick* and *Gandhi* in the *Civilization IV* [1]. Currently it is not rare that intelligent NPCs may defeat experienced human players. However, the NPCs' actions tend to appear artificial after some time of playing, and rule out any surprise that human opponents would provide. This has motivated a line of research on constructing believable NPCs in interactive games [4,6,20,21].

Much of the existing research takes the macro-perspective on designing believable behavior. For example, the *Soar* architecture provides a cognitive model to develop believable agents in computer games [9,19]. In parallel, the *ICARUS* framework facilitates the development of goal-directed agents in games [10]. The *Emotivector* model encodes an anticipatory mechanism for the believability enhancement on designing human-like characters [11]. While the mentioned research significantly drives the study on believable behavior, it requires much effort to integrate the associated frameworks or models into the routine design of NPCs in game productions. In this paper, we will adapt behavior trees [7,3]

L. Cao et al.: ADMI 2012, LNAI 7607, pp. 81–92, 2013.

- a new generation of script language for game design - to develop believable behavior models of NPCs.

The notion of believability has been studied in the fields of arts, psychology and computer science, for a couple of decades [2,13]. Linking to NPCs in computer games, believable behavior is not especially smart and is coupled with some unpredictability [14,17]. In other words, behaviors that are too intelligent will be rapidly categorized as being unreal, and that are very unpredictable may lead to the feeling of randomness. We need to make a proper tradeoff between the intelligence and randomness of the behavior. According to this spirit, the objective on designing believable behavior is to achieve the intelligence of NPCs' behavior and simultaneously maintain the diversity of their behavior. We will formulate the believability design as one multi-objective optimization problem and compute its optimal solutions if they exist.

We focus on the realization of believable behavior based on behavior trees. As behavior trees plan NPCs' actions in the game-play we may construct the optimal behavior through AI planning and learning techniques [15]. To make NPCs act intelligently, we let their behavior approach the optimal one. We use a probability-based distance measurement to quantify the intelligence of NPCs' behavior. Meanwhile, we use the information entropy [5] to measure the diversity of NPCs' behavior. The combination of these two measurements provides a quantitative approach to formulate the believability of NPCs' behavior. The formulation allows us to compute the best believability. Finally, we evaluate the believability design in both simulations and user tests, and demonstrate the practical utility of our techniques.

2 Background: Behavior Tree

Behavior tree is a graphical representation for structuring NPCs' behavior in a modular manner so that both designers and programmers can work together in the game development [7,3]. It starts with a *Root* node (denoted by a down triangle shape) and normally ends with an *Action* node (denoted by a circle shape) as its leaf. We show the other three types of basic nodes in Fig. 1. We refer the reader to [7,3] for more details on the representation of behavior trees.

A *Sequence* node has one or more *Action* nodes as its children, and is used whenever a sequence of actions have dependency upon one another. The *Sequence* executes the first action ($A1$), and if the execution returns success it continues the next one ($A2$) and so on. Failure of any action terminates the execution of the *Sequence* node. A *Selector* node contains a set of independent *Action* nodes, and may choose one of them for an execution. The *Selector* fails only if neither of its children nodes can be successfully executed. In general, game designers assign a probability distribution, $(p_{A_1}, \cdots, p_{A_N})$, over the set of N actions. The *Selector* executes one of the actions according to its probability distribution. A *Decorator* node is inserted on the top of an action node or a subtree in order to provide additional functionalities to a generic behavior. For example, one type of *Decorator* can either limit the number of times that

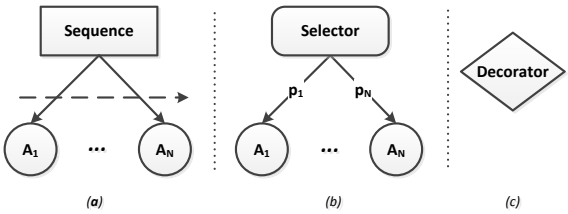

Fig. 1. (*a*) A *Sequence* node with N actions $(A_1, \cdots, A_N)$ where the follow of execution is prescribed by the dotted arc; (*b*) A *Selector* node with N actions any of which can be selected; (*c*) A *Decorator* node adds additional functionalities to behavior

the subtree could be called or retrieve the status message from the execution of the associated actions. Since the *Decorator* is often transformed into some associated properties of *Action* nodes, we focus on a *canonical* behavior tree that mainly contains the *Sequence*, *Selector* and *Action* nodes. Here we take the popular 2-D fighting game (implemented in the **MUGEN** game engine[1]) for one example of behavior trees.

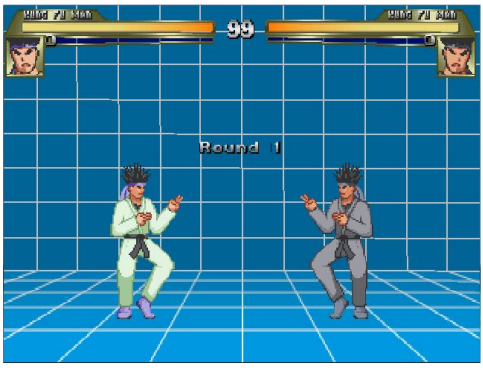

Fig. 2. A human player is fighting with a *Kung Fu Man* in the **MUGEN** game. The yellow bars above show their health points.

In Fig. 2, the **MUGEN** game hosts two players: one NPC (called *Kung Fu Man*) and one human player. The stage is a 2-D arena where the players can move freely horizontally, and any movement in the vertical axis is achieved by either a jump or a crouching move. The behavior tree as designed in Fig. 3 commands the actions of *Kung Fu Man* in the game-play.

Example 1 (Behavior Tree). *When the game starts, the* Kung Fu Man *chooses either Attack or Defend on executing the Selector node, Choose Mode.*

[1] `http://www.elecbyte.com/`

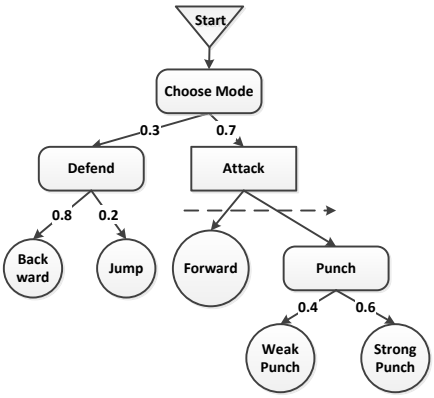

Fig. 3. A behavior tree is designed for an aggressive *Kung Fu Man*

*He is a bit aggressive since there is a large probability (*0.7*) that he attacks a human player. Assume that he selects the Attack mode, he needs to execute a sequence of actions as indicated by the Sequence node, Attack. He will firstly move Forward, and then execute the action Punch if he succeeds in the movement. Finally, he will attack the opponent using one of two punch types. As the probabilities specified in the Selector node, Punch, he may launch a heavy attack with the Strong Punch (that has a larger probability 0.6). The behavior tree returns a success if the* Kung Fu Man *does not fail to execute the actions. Subsequently, a new traversal of the tree will be initiated (at the root node) to control his actions.*

By linking *Selector* and *Sequence* nodes, a behavior tree prescribes a sequence of actions for NPCs in computer games. The configurations of the tree structure (connections of nodes) and parameters (probabilities in the *Selector* node) directly control the NPCs' behavior. In this paper, we mainly exploit the parameter settings for the purpose of designing believable behavior.

3 Believable Behavior Design

Behavior trees provide a simple, scalable and modular solution to design complex NPCs' behavior in computer games. The model settings offer a flexible mechanism to control the behavioral dynamism. By configuring the action probabilities, we expect NPCs to exhibit believable behavior in the game-play. We will firstly formulate the believability of NPCs' behavior as one computational design and analyze its solutions afterwards.

3.1 Computational Believability

A behavior tree structures the relations of action nodes and imposes a set of parameters over actions in a *Selector* node. Formally, we define a behavior tree below.

Definition 1 (Behavior Tree). *Define a behavior tree as:* $\mathcal{T} = \langle \mathcal{V}, \mathcal{E}, \mathcal{P} \rangle$ *where* $\mathcal{V}$ *is a set of Action, Sequence, and Selector nodes;* $\mathcal{E}$ *is a set of edges connecting the nodes; and* $\mathcal{P}$ *is a set of probability distributions guarding the edges in the Selector nodes.*

We assume the known structure of behavior trees including $\mathcal{V}$ and $\mathcal{E}$, and will find a proper setting of $\mathcal{P}$ for designing believable behavior. For the probability distribution, $\mathcal{P}$, we further denote it as $\mathcal{P} = \langle \mathcal{P}_{S_1}, \cdots, \mathcal{P}_{S_M} \rangle$ where $\mathcal{P}_{S_i} = (p_{A_1}, \cdots, p_{A_N})$ is a probability distribution over N actions under the *Selector* node, S_i.

Given a behavior tree $\mathcal{T}$, we may retrieve a set of behavior paths from the tree. Each path is a sequence of actions that an NPC will experience in the game-play. Formally, let $\mathcal{H} = \langle \mathcal{H}_1, \cdots, \mathcal{H}_R \rangle$ be the set of behavior paths where $\mathcal{H}_j = (A_1, \cdots, A_K)$ is a sequence of actions. We define the probability of a path $\mathcal{H}_j$ as:

$$P(\mathcal{H}_j) = \Pi_{k=1}^{K} p(A_k) \tag{1}$$

Here A_k refers to the action labeled either in the *Action* node, or in the *Sequence* node, or in the *Selector* node. Its probability depends on the type of its parent. Consequently, $p(A_k) = 1$ if A_k is a child of either the *Sequence* node or the *Root* node; otherwise, $p(A_k)$ is equal to the probability p_{A_k} as defined for A_k under the corresponding *Selector* node.

Example 2 (Behavior Path). *Given the behavior tree in Fig. 3, we can get 4 behavior paths as:* $\mathcal{H} =< (ChooseMode, Defend, Backward), (ChooseMode, Defend, Jump), (ChooseMode, Attack, Forward, Punch, WeakPunch), (ChooseMode, Attack, Forward, Punch, StrongPunch)$. *The probability of* $\mathcal{H}_3$ *is computed as:* $P(\mathcal{H}_3) = 1 \times 0.7 \times 1 \times 1 \times 0.4 = 0.28$. *The probabilities for all paths are:* $P(\mathcal{H}) = (0.24, 0.06, 0.28, 0.42)$.

Behavior paths, together with the probabilities, specify how an NPC shall act in the game world. The NPC acts intelligently if it is able to learn from its experience. In other words, we can develop intelligent behavior for the NPC by automatically learning its behavior trees. As the set of paths are known given the structure of behavior trees, we need to learn the probabilities, $P(\mathcal{H})$, from game experience. A set of AI/statistical learning techniques have been adapted for this purpose, which is one of the main focuses of AI research in computer games [12]. We may resort to similar techniques that result in the optimal probability values, $P^*(\mathcal{H})$, for intelligent behavior of an NPC.

The probability setting, $P^*(\mathcal{H})$, generates the most intelligent behavior for NPCs since the probabilities are learned from the NPC's experience. The question is: how to measure the intelligence of the NPC's behavior if the NPC executes the behavior with a different probability setting of $P(\mathcal{H})$. Instead of providing a direct measurement, we gauge how its intelligence approaches that of the NPC with the setting of $P^*(\mathcal{H})$. To measure the intelligence gap, we use the distance between

two probability distributions, $P^*(\mathcal{H})$ and $P(\mathcal{H})$. Formally, we use the Kullback-Leibler (KL) divergence [8] as defined below.

$$D_{KL}[P(\mathcal{H})||P^*(\mathcal{H})] = \sum_j P(\mathcal{H}_j) ln \frac{P(\mathcal{H}_j)}{P^*(\mathcal{H}_j)} \tag{2}$$

In order to design the intelligent behavior, we need to minimize the distance, D_{KL}, in Eq. 2. The distance converges to zero when the behavior of an NPC achieves the highest intelligence in the optimal setting of $P^*(\mathcal{H})$.

On the other hand, we expect an NPC to execute a broad set of behavior in the game-play. We choose Shannon entropy [5] as the measurement of the behavior diversity. The diversity of the behavior with the probability distribution, $P(\mathcal{H})$, is defined in Eq. 3. A large entropy value indicates more types of actions that an NPC will perform in the real play. Consequently, the NPC's behavior becomes more unpredictable from the eyes of its opponents.

$$E[P(\mathcal{H})] = -\sum_j P(\mathcal{H}_j) ln P(\mathcal{H}_j) \tag{3}$$

As we mentioned, the believability seeks for a good balance of the intelligence and the diversity of NPCs' behavior. The design of believable behavior is to achieve the optimal solutions of the intelligent behavior while to maintain the diversity of the behavior. In other words, we will minimize the KL divergence between $P(\mathcal{H})$ and the optimal one $P^*(\mathcal{H})$, and simultaneously maximize the information entropy of $P(\mathcal{H})$. Formally, we aim to compute the probability distributions, $P(\mathcal{H})$, that are solutions to the optimization problem below.

$$\begin{aligned} &\textbf{Objective} : max \\ &\quad BEL = -K_1 \textstyle\sum_j P(\mathcal{H}_j) ln P(\mathcal{H}_j) - K_2 \sum_j P(\mathcal{H}_j) ln \frac{P(\mathcal{H}_j)}{P^*(\mathcal{H}_j)} \\ &\textbf{Variables} : P(\mathcal{H}) = \langle P(\mathcal{H}_1), \cdots, P(\mathcal{H}_R) \rangle \\ &\textbf{Constraints} : \textstyle\sum_j P(\mathcal{H}_j) = 1 \end{aligned} \tag{4}$$

where $P^*(\mathcal{H})$ are the optimal solutions of intelligent behavior, and K_1 and K_2 are the positive values weighting the diversity and intelligence of behavior respectively in the believability function, BEL.

We compute the solutions, $P(\mathcal{H})$, by applying the partial derivative in the objective function, BEL. Accordingly, the design achieves the optimal believability where an NPC executes the behavior path, $\mathcal{H}_j$, with the probability in Eq. 5.

$$P(\mathcal{H}_j) = \frac{P^*(\mathcal{H}_j)^{\frac{K_2}{K_1+K_2}}}{\sum_j P^*(\mathcal{H}_j)^{\frac{K_2}{K_1+K_2}}} \tag{5}$$

We note that the the probabilities of believable behavior depend on weights between the intelligence and the diversity. An NPC behaves randomly ($P(\mathcal{H}_j) = \frac{1}{R}$) if $K_2 \ll K_1$, and shows the highest intelligence ($P(\mathcal{H}_j) = P^*(\mathcal{H}_j)$) if $K_2 \gg K_1$.

Example 3 (Believability Solutions). *Given a set of 2 behavior paths,* $\mathcal{H} =< \mathcal{H}_1, \mathcal{H}_2 >$*, we learn the path probabilities,* $P^*(\mathcal{H})=(0.3, 0.7)$*, for intelligent behavior from game data that record the NPC's performance. We plot the believability function, BEL, for different settings of* K_1 *and* K_2 *in Fig. 4. Selections of* (K_1, K_2) *values balance the factors that contribute into the believability design. Fig. 4(a) favors the intelligence as the dominating attribute of the believability. Hence its solution,* $P(\mathcal{H})=(0.337, 0.663)$*, approaches the learned probabilities. On the other hand, Fig. 4(c) attributes the believability to the diversity of actions, and its design,* $P(\mathcal{H})=(0.479, 0.521)$*, is close to the random behavior. This may happen to an insane NPC in games. Note that the BEL values are not comparable across different* (K_1, K_2)*.*

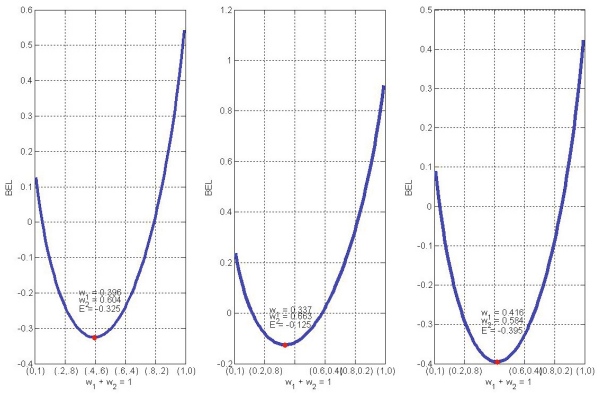

Fig. 4. The probabilities (denoted by red triangles) of believable behavior given K_1 and K_2 values that weight the diversity and the intelligence of behavior respectively

Once we get the path probabilities, $P(\mathcal{H})$, we can compute the probabilities, p_{A_k}, for actions through Eq. 1. This completes the parameter setting of $\mathcal{P}$ for the believable behavior design in behavior trees .

3.2 Bottom-Up Design

As a hierarchical model, a behavior tree is goal-oriented and allows the recursive implementation of complex NPCs' behavior. In practice, it firstly defines a high-level goal (in a *Root* node) which it attempts to achieve, and then links to a set of sub-goals until it constructs primitive actions at the leaves of the tree. Consequently, the tree may become deeply nested and is prone to be asymmetric. Computing the global believability, $P(\mathcal{H})$, for an entire tree involves difficulty in learning the probabilities of intelligent behavior, $P^*(\mathcal{H})$, for a large set of behavior paths.

Following the spirit of modular design in behavior trees, we may compute the local believability, $P_{\mathcal{T}_l}(\mathcal{H})$, for a set of subtrees and incrementally complete the believability design of $\mathcal{T}$ ($=\cup_l \mathcal{T}_l$). To make a further step, we may arrange the subtrees in a hierarchical way and recursively compute the believability in a bottom-up manner.

Let $\mathcal{T}_l$ be level l ($l \in [1, L]$) subtree in $\mathcal{T}$ (the root of $\mathcal{T}$ is in level 0), and $BEL_{\mathcal{T}}$ ($BEL_{\mathcal{T}_l}$) be the global (local) believability function respectively. According to the believability definition, BEL, in Eq. 4, we derive the relations between the global and local believability functions in Eq. 6. It shows that the global believability is the sum of a set of weighted local believability.

$$\begin{aligned} BEL_{\mathcal{T}} = BEL_{\mathcal{T}_1} + P(\mathcal{T}_2)[BEL_{\mathcal{T}_2} + \cdots + \\ P(\mathcal{T}_l)[BEL_{\mathcal{T}_l} + \cdots + P(\mathcal{T}_L)[BEL_{\mathcal{T}_L}]\cdots]\cdots] \end{aligned} \tag{6}$$

where $P(\mathcal{T}_l)$ is the probability of the root node, A_k, in the subtree $\mathcal{T}_l$. The probability is equal to p_{A_k} if the parent of A_k is a *Selector* node in $\mathcal{T}_{l-1}$; otherwise, it is equal to $p_{A'_k}$ where A'_k is the closest ancestor of A_k and is a child of a *Selector* node.

As $P(\mathcal{T}_l)$ is one of the probability parameters in $\mathcal{T}_{l-1}$, solutions to maximize the believability function, $BEL_{\mathcal{T}}$, can be achieved by computing the local believability sequentially from $BEL_{\mathcal{T}_L}$ to $BEL_{\mathcal{T}_1}$. Eq. 7 shows that the optimal solution to the global believability, $P(\mathcal{H})=<(P_1(\mathcal{H}), \cdots, P_l(\mathcal{H}), \cdots, P_L(\mathcal{H})>$, can be distributed over the local believability for the set of hierarchical subtrees where $P_l(\mathcal{H})$ is the optimal solution to the believability design in level l subtree, $\mathcal{T}_l$.

$$\begin{aligned} \max_{P(\mathcal{H})} BEL_{\mathcal{T}} = \max_{P_1(\mathcal{H})} [BEL_{\mathcal{T}_1} + P(\mathcal{T}_2) \max_{P_2(\mathcal{H})} [BEL_{\mathcal{T}_2} \\ + \cdots + P(\mathcal{T}_l) \max_{P_l(\mathcal{H})} [BEL_{\mathcal{T}_l} + \cdots \\ + P(\mathcal{T}_L) \max_{P_L(\mathcal{H})} [BEL_{\mathcal{T}_L}]\cdots]\cdots]] \end{aligned} \tag{7}$$

We shall note that the believability design of level l subtree, $\mathcal{T}_l$, needs to find the optimal solutions to maximize the function in Eq. 8. We may show that the analytic solutions, $P_l(\mathcal{H})$, still enjoy the close form similarly in Eq. 5.

$$\max_{P_l(\mathcal{H})} [BEL_{\mathcal{T}_l} + P(\mathcal{T}_{l+1}) OPT(BEL_{\mathcal{T}_{l+1}})] \tag{8}$$

where $OPT(BEL_{\mathcal{T}_{l+1}})$ is the optimal believability value generated from the believability design of $\mathcal{T}_{l+1}$.

In summary, we may decompose the global believability optimization problem into a set of local optimization problems and still achieve the same solutions to the believability in a bottom-up design. The statement is given in Theorem 1.

Theorem 1 (Design Optimality). *Bottom-up design preserves the optimal believability of the global design for an entire behavior tree.*

4 Evaluation and User-Study

We experimented the believability design in the aforementioned **MUGEN** game [2]. We used the *N-Gram* statistical models [18] to learn the probability of intelligent behavior, $P^*(\mathcal{H})$, and computed the probability of believable behavior given different settings of (K_1, K_2) in Eqs. 5 and 1. The resulting probabilities, $p(A_k)$, were used to configure the parameters, $\mathcal{P}$, of the behavior trees. During the game-play, NPCs are controlled by the associated behavior trees. We use NPC(K_1, K_2) to denote the NPC that displays believable behavior given one setting of (K_1, K_2). We report the NPCs' performance when they compete with either other NPCs or human-players. In addition, we invite human players to rank the NPCs in terms of the intelligence and believability of their behavior and advise proper values of (K_1, K_2) in relevant games.

By extending the behavior tree in Fig. 3, we developed five NPCs listed as: NPC$_1(0, 1)$, NPC$_2(0.15, 0.85)$, NPC$_3(0.25, 0.75)$, NPC$_4(0.4, 0.6)$, and NPC$_5(0.5, 0.5)$. Note that the NPC$_1(0, 1)$ is configured with the learned probability, $P^*(\mathcal{H})$, for the believable behavior. In addition, we designed three stereotypes of NPCs (NPC$_A$: *Aggressive*, NPC$_N$:*Neutral*, and NPC$_D$:*Defensive*) that represent typical roles in the **MUGEN** games. The NPCs differ in the probabilities assigned to actions under the *Selector* nodes. We let NPC$_i$ (i=1,$\cdots$,5) start with random actions and compete with the stereotypes individually over 200 matches. During the competition, we had the NPC$_i$ learn from the experience every 20 matches and designed its behavior based on the new probability, $P^*(\mathcal{H})$. We report the total number of matches that the NPC$_i$ won over every stereotype in Table 1.

Table 1. The NPCs (NPC$_1$-NPC$_5$) learn to compete with their opponents (NPC$_A$-NPC$_D$). The more intelligence (NPC$_1$ $>\cdots>$ NPC$_5$) the more matches the NPCs win.

Matches	**NPC$_1$**	**NPC$_2$**	**NPC$_3$**	**NPC$_4$**	**NPC$_5$**
NPC$_A$	**181**	176	163	118	72
NPC$_N$	**182**	179	170	127	83
NPC$_D$	**186**	180	170	135	98

Table 1 shows that the NPCs (NPC$_1$-NPC$_5$) perform intelligently when they assimilate most of their learning results given the setting $K_2 > K_1$. They lose few matches if they have fully exploited (where K_2=1) the behavior of their opponents. The results also demonstrate the utility of the *N-Gram* techniques on learning the NPCs' behavior in the game.

We enrolled 27 participants to observe the matches and rank both the intelligence and believability of the NPCs (NPC$_1$-NPC$_5$) when the NPCs were playing

[2] Due to the limited space, we show only the evaluation on the **MUGEN** game while we also conducted study in the popular **StarCraft** Game.

with their opponents. Most of the participants have some experience on the **MUGEN** game. The criteria that they used to evaluate the believability were mainly on plausible sequences of attacks, diverse behavior, and predictable actions. We report the average rankings (with standard deviation) of the NPCs in Table 2. As expected, the $\text{NPC}_1(0, 1)$ was ranked as the most intelligent one over all competitions with different types of opponents. However, it lost to $\text{NPC}_2(0.15, 0.85)$ on the aspect of the believability.

Table 2. Average rankings of the NPCs (NPC_1-NPC_5) in terms of the intelligence and the believability. 5 is the highest and 1 is the lowest.

NPCs	**Criteria**	$\textbf{NPC}_A$	$\textbf{NPC}_N$	$\textbf{NPC}_D$
$\textbf{NPC}_1$	Intelligence	**4.63(0.50)**	**4.38(0.5)**	**4.19(0.66)**
	Believability	3.31(1.14)	3.06(1.44)	2.75(1.34)
$\textbf{NPC}_2$	Intelligence	4.19(0.75)	4.13(0.96)	4.13(1.02)
	Believability	**4.38(0.88)**	**4.25(0.77)**	**4.06(1.06)**
$\textbf{NPC}_3$	Intelligence	3.13(0.5)	3.38(0.89)	3.38(1.02)
	Believability	3.44(1.03)	3.88(0.89)	3.94(0.77)
$\textbf{NPC}_4$	Intelligence	2.06(0.25)	2.13(0.34)	2.31(0.60)
	Believability	2.25(0.77)	2.06(0.25)	2.50(0.97)
$\textbf{NPC}_5$	Intelligence	1(0)	1(0)	1(0)
	Believability	1.69(1.49)	1.81(1.52)	1.75(1.39)

We made a further step to compare pairs of rankings through t-tests. Table 3 shows the p-values of the tests between the average rankings of the NPCs. It is a bit surprising that the $\text{NPC}_1(0, 1)$ was not perceived as being significantly smarter than the $\text{NPC}_2(0.15, 0.85)$. This indicates that the solutions (where K_2=0.85) are sufficient to exhibit the intelligent behavior. Additional diversity of the behavior does not compromise the intelligence, but generate the desired believability in most cases.

Table 3. p-values from t-tests on the pair comparisons. Entries with an underline are significant at the 95% confidence level.

Criteria	**NPCs**	$\textbf{NPC}_A$	$\textbf{NPC}_N$	$\textbf{NPC}_D$
Intelligence	$\text{NPC}_1 > \text{NPC}_2$	0.08	0.23	0.43
	$\text{NPC}_2 > \text{NPC}_3$	<u>0.00</u>	<u>0.05</u>	0.06
	$\text{NPC}_3 > \text{NPC}_4$	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>
	$\text{NPC}_2 > \text{NPC}_4$	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>
Believability	$\text{NPC}_2 > \text{NPC}_1$	<u>0.00</u>	<u>0.02</u>	<u>0.01</u>
	$\text{NPC}_2 > \text{NPC}_3$	<u>0.01</u>	0.11	0.37
	$\text{NPC}_3 > \text{NPC}_4$	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>
	$\text{NPC}_2 > \text{NPC}_4$	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>

Table 4. Average rankings of the NPCs (NPC_1-NPC_3) in the real-play. 3 is the highest and 1 is the lowest.

Criteria	$\mathbf{NPC_1}$	$\mathbf{NPC_2}$	$\mathbf{NPC_3}$
Intelligence	**2.58(0.51)**	2.25(0.75)	1.17(0.39)
Believability	1.58(0.67)	**2.75(0.62)**	1.67(0.65)

We invited 18 out of 27 participants to play with the top three NPCs (NPC_1-NPC_3) over 100 matches. Subsequently the participants ranked the NPCs according to their personal game experience. In Table 4, the results are consistent with the analysis above. The believability of the NPC_2's behavior is significantly better than that of the others. Most of the participants felt uncomfortable when the $NPC_1(0, 1)$ launched non-breaking attacks with strong punches. It was convincing that the $NPC_2(0.15, 0.85)$ took a light jump after some punches.

5 Conclusion

We propose a computational model for game designers that allow them to create believable behavior for NPCs in computer games. The believability model is rooted in a generic representation of behavior trees and sophisticated AI techniques. We quantify the believability by measuring the intelligence and diversity of NPCs' behavior. In principle, the believability design is seeking for a balance of these two measurements. We further formulate the believability design as one optimization problem and provide analytic solutions to the optimal believability. More importantly, we observe that the bottom-up design can guarantee the optimal believability of the entire behavior tree. This facilitates the practical development on designing believable behavior.

The computational model considers two important attributes (intelligence and diversity of behavior) in the believability design. For future work, we will explore more factors that may contribute into the believability of NPCs' behavior. The challenge is on the development of a quantitative measurement for the attributes. We are more interested in integrating the additional attributes into the established model.

References

1. Amato, C., Shani, G.: High-level reinforcement learning in strategy games. In: Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent (AAMAS), pp. 75–82 (2010)
2. Bates, J.: Virtual reality, art and entertainment. Presence 1(1), 133–138 (1992)
3. Champandard, A.J.: Behavior trees for next-gen game ai. Tutorial, AiGameDev.com (2008)
4. Chang, Y., Maheswaran, R., Levinboim, T., Rajan, V.: Learning and evaluating human-like npc behaviors in dynamic games. In: Proceedings of the Seventh Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE), pp. 8–13 (2011)

5. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley-Interscience, New York (1991)
6. Doirado, E., Martinho, C.: I mean it!: detecting user intentions to create believable behaviour for virtual agents in games. In: Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent (AAMAS), pp. 83–90 (2010)
7. Isla, D.: Handling complexity in the halo 2 ai. In: Proceedings of the Fifteenth Conference on Game Developers Conference (2005)
8. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Statist. 22(1), 79–86 (1951)
9. Laird, J.E., Newell, A., Rosenbloom, P.S.: Soar: An architecture for general intelligence. Artificial Intelligence 33(1), 1–64 (1987)
10. Langley, P., Choi, D.: A unified cognitive architecture for physical agents. In: Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence (AAAI), pp. 876–881 (2006)
11. Martinho, C., Paiva, A.: Using anticipation to create believable behaviour. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI), pp. 175–180 (2006)
12. Rabin, S.: AI Game Programming Wisdom 4. Course Technology (2009)
13. Scott Neal Reilly, W.: Believable Social and Emotional Agents. PhD thesis, School of Computer Science, Carnegie Mellon University (1996)
14. Riedl, M.O., Stern, A.: Believable agents and intelligent scenario direction for social and cultural leadership training. In: Proceedings of the Fifteenth Conference on Behavior Representation in Modeling and Simulation (2006)
15. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice-Hall (2003)
16. Tan, C.T., Cheng, H.: Implant: An integrated mdp and pomdp learning agent for adaptive games. In: Proceedings of the Fifth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE), pp. 94–99 (2009)
17. Tence, F., Buche, C., De Loor, P., Marc, O.: The challenge of believability in video games: Definitions, agents models and imitation learning. CoRR abs/1009.0451 (2010)
18. Witten, I.H., Bell, T.C.: The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. IEEE Transactions on Information Theory 37(4), 1085–1094 (1991)
19. Xu, J.Z., Laird, J.E.: Combining learned discrete and continuous action models. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI), pp. 1449–1454 (2011)
20. Zeng, Y., Buus, D.P., Hernandez, J.C.: Multiagent based construction for human-like architecture. In: Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2007), pp. 409–411 (2007)
21. Zeng, Y., Hernandez, J.C., Buus, D.P.: Swarmarchitect: a swarm framework for collaborative construction. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2007), pp. 186–186 (2007)