Towards A Data-driven Approach to Scenario Generation for Serious Games

Abstract

Serious games have recently shown great potential to be adopted in many applications, such as training and education. However, one critical challenge in developing serious games is the authoring of a large set of scenarios for different training objectives. In this paper, we propose a data-driven approach to automatically generate scenarios for serious games. Compared to other scenario generation methods, our approach leverages on the simulated player performance data to construct the scenario evaluation function for scenario generation. To collect the player performance data, an AI player model is designed to imitate how a human player behaves when playing scenarios. The AI players are used to replace human players for data collection. The experiment results show that our data-driven approach provides good prediction accuracy on scenario's training intensities. It also outperforms our previous heuristic-based approach in its capability of generating scenarios that match closer to specified target player performance.

Keywords: scenario generation, serious games, virtual agents

1 Introduction

With the rapid advances in computer animation and virtual agent technologies, designing games and virtual worlds to support learning and skill acquisition is gaining tremendous momentum. Serious games, which have useful purposes other than entertainment, have been applied to many applications, such as training [1, 2, 3] and education [4, 5]. Compared to traditional training and tutoring methods, serious gaming has the advantage of offering an engaging, interactive and collaborative virtual experience, which can lead to better learning performance [5].

In serious games for training, one of the key requirements to ensure effective learning is the provision of scenarios, which describe the flow of events during game-play. By playing different scenarios, a trainee can exercise different skills or undertake certain missions. Thus, it is essential to design the scenarios that can meet different training objectives and, at the same time, be customized for individual trainees. However, manual authoring of scenarios is a time-consuming and tedious process and it has become a critical bottleneck to training [6]. Our goal, therefore, is to design a scenario generation system that can automatically generate scenarios while allowing a trainer to have a directed control over the generation process.

Our previous work [7, 8] has proposed a genetic algorithm (GA)-based scenario generation system, which considers both the trainer's preferences and trainee's skill levels in generation process. The system relies on a heuristic-based fitness evaluation, which examines how well (in terms of training intensities) a given scenario can match with trainer' inputs. In this paper, we extend our work by designing a data-driven approach for scenario evaluation. The data-driven approach can provide a more accurate approach for scenario evaluation, as the evaluation function is constructed based on the data collected on the actual effect of a scenario in simulation. It is also more flexible to adapt to new types of scenarios, as an automated data training process is employed.

The key contributions of our data-driven approach described in this paper are as follows. First, an artificial neural network (ANN) based data training process is designed to automatically train ANNs based on the data collected from simulation. The trained ANNs are used for evaluating how well a scenario matches with the training objectives set by trainer. Second, an AI player model is designed and utilized for collecting player performance data in the data collection process. The AI player model is designed to imitate how a human player behaves when playing a scenario. The use of the AI player can facilitate the large-scale data collection without having human players play the scenarios. Lastly, a scenario generation framework based on the data-driven approach is introduced and evaluated in a military operation training game.

2 Related Work

The problem of scenario generation broadly pertains to the content generation issue in developing games and interactive virtual environments. In recent years, various techniques have been explored for automatically generating content of different types, such as agent behaviors [9, 10], game levels and maps [11, 12], and narratives [13, 14]. Compared to other types of content generation, research in the generation of scenario (i.e., the flow of events) is still an emerging area with a limited number of approaches being proposed.

The research works on offline scenario generation have focused on how to generate globally optimized/customized scenarios given different objectives. For example, Hullett and Mateas [15] described a HTN planning-based generation system to generate scenarios from the selected pedagogical goals. Martin et al. [16] used functional L-systems to specify generation rules to create scenario elements. Zook et al. [6] modeled the scenario generation as a combinatorial optimization process and applied genetic algorithm to search for scenarios that maximize a set of evaluation criteria.

Our work also focuses on the offline scenario generation. But, it differs from the above work not only in the generation methodology, but also in the evaluation of scenarios. Our scenario generation system considers how multiple mission objectives can be exercised in a single scenario. Besides, the previous approaches use either designer-defined goals or heuristics to govern the generation process. In contrast, our system adopts a data-driven approach to evaluating a scenario with respect to its effects on different mission objectives. Thus, it does not rely on designer's intuition to derive the mapping between a scenario and its fitness.

Some data-driven methods have been applied to generate content for games. Shaker et al. [11] applied a data-driven player experience model for generating game levels of *Super Mario Bros* game using neural networks. Yannakakis et al. [17] proposed a mechanism to generate camera profiles for in-game camera control based on the player's affective states models. The models are trained using neuroevolutionary preference learning on questionnaire data of players.

Compared to these works, our data-driven approach does not require collecting data on real human players. Instead, we design a simulation-based data collection mechanism, which uses an artificial agent (i.e., AI player) to play through the scenarios being evaluated. The AI player is designed and validated to accurately reflect real player behaviour in our training game. The use of the AI player can help to reduce the cost for involving human players for data collection.

3 Testbed Training Game

As a testbed for our studies, we have developed a food distribution training game (see Figure 1). The game is created using DI-Guy[1], which is a state-of-art human simulation software for virtual training. The developed game simulates a food distribution mission in a post-disaster area, where hundreds of civilians are queuing for food and water supplies. Figure 1 shows the layout of the environment with color-coded areas. The environment is composed of the four main areas: the waiting area for new civilians (green, top-right), the first part of the queue (grey, bottom-right), the second part of the queue (yellow, bottom-left) and the distribution area (brown, top-left). Each area is guarded by an infantry force of several soldiers. In or-

[1]DI-Guy: http://www.diguy.com/diguy/

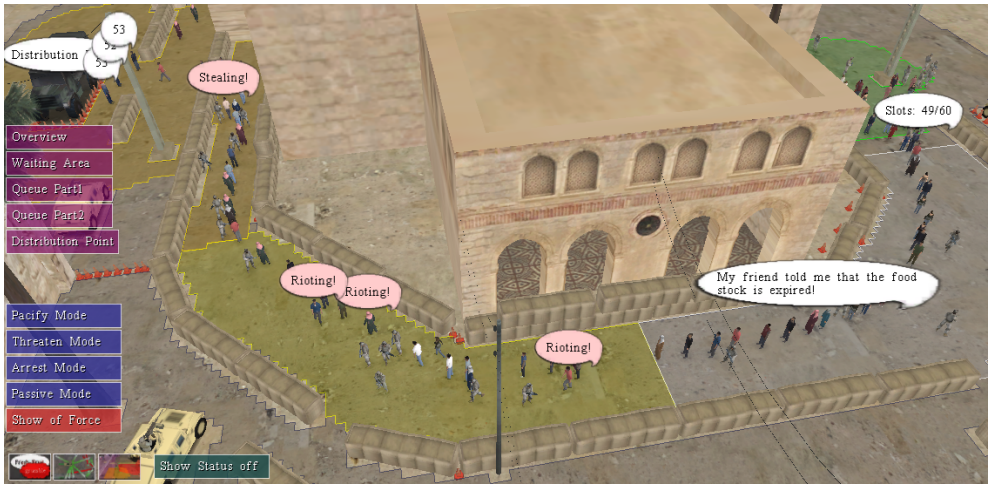

Figure 1: Screenshot of the food distribution training game.

der to issue commands, a trainee has to switch to one particular area (i.e., a zoom-in view) by clicking one of the area buttons (the purple buttons on the left side of Figure 1).

The training game is designed to practice the trainee's ability to achieve a set of mission objectives (MOs). Each MO specifies certain mission-specific task or ability to be practiced by trainee. In our current food distribution game, the mission objectives are defined as: MO_1 - to control the crowd anger level, MO_2 - to identify instigators, and MO_3 - to prevent civilians from stealing food. In order to exercise these defined MOs, different types of events are designed and implemented. Table 1 gives a list of event types that we have realized in the current game and their contributions to different MOs. These events are used to form scenarios as we will describe in next section.

During the game-play, a trainee has to react to the crowd violence (i.e., rioting and stealing), caused by the events being injected into the game. To control the violent crowd, a trainee has to promptly react to the changing situation and issue proper commands. A command is issued by right-clicking on an agent and selecting a tactical command to execute. Once a tactical command is selected, a nearby soldier agent will be assigned to execute the command. In our game, a trainee can issue three types of tactical commands: the *pacify* command, which attempts to calm down an agent; the *threaten* command, which stops an agent from performing stealing behavior; and the *arrest* command, which is used to arrest an instigator agent and escort it to a secure detainment vehicle.

4 Data-driven Framework for Scenario Generation

In order to conduct effective training, it is necessary to generate a variety of scenarios (i.e., flow of events) that can meet different training objectives and at the same time be customized for individuals. In our previous work [7, 8], we have proposed a genetic algorithm (GA)-based scenario generation system, which can automatically generate training scenarios based on trainer's preferences and trainee's skill levels. This work extends previous work by adopting a data-driven approach to constructing the fitness function of GA for scenario evaluation. The artificial neural network (ANN) model is used to approximate the mapping between the input scenarios and scenarios' training intensities.

Figure 2 presents an overview of our data-driven scenario generation framework. The framework comprises three main modules: *content design*, *scenario generation* and *data training*. The *content design* module supports scenario designer in the creation of scenario content. Here, all the domain-specific content (i.e., mission objectives, event types and instances) are created. The *data training* module is responsible for training the ANN based on the data collected from the simulation. After training, the trained ANN is incorporated into the GA-

Table 1: List of event types.

Event Type	Contributed MOs	Event Description
Instigation	MO_1, MO_2	Inject instigator agents into a specific area. Instigators will move randomly around the area and try to make crowd angry purposely.
Violence	MO_1, MO_3	Inject violent civilian agents into a specific area. Compared to normal civilians, violent civilians' riot and steal behaviors can be triggered more easily.
Rumor spreading	MO_1	Trigger a normal civilian agent to perform rumor spread behavior. The rumor will be spread among nearby agents and cause massive crowd anger if not controlled.
Jump queue	MO_1	Trigger civilian agents in the queue to perform jump queue behaviors. The agents who are affected by queue jumping may become angry.
Unguarded	MO_3	Temporarily remove the guarding soldiers at the distribution points for a period of time. This increases the civilian agent's desire to steal. Note that the guarding soldiers are not affected by trainee's commands

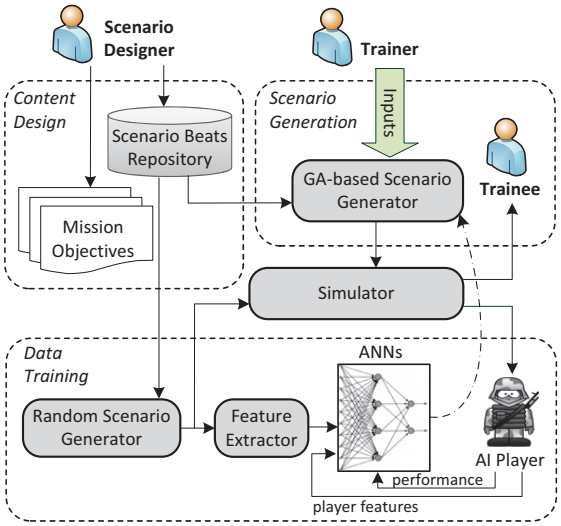

Figure 2: Overview of data-driven scenario generation framework.

based scenario generator in the *scenario generation* module for evaluating scenario fitness. The *scenario generation* module is then used to let a trainer generate the desired scenarios based on his/her preferences and send the scenarios to the simulator for training. We detail each module in the following subsections.

4.1 Content Design

To generate different scenarios for a particular mission type (e.g., food distribution), a scenario designer needs to first define the mission objectives (MOs) to be exercised. For example, in our food distribution mission, three mission objectives are defined (see section 3). The defined MOs determine *what* to be trained in a given mission. In order to exercise different MOs, the scenario designer needs to define different types of events to be simulated (e.g., as in Table 1). Each type of event can contribute to the exercise of different MOs. For example, the instigation event in Table 1 can be used for practicing both MO_1 and MO_2 as defined in section 3. Each event type is characterized by certain parameter(s). By assigning value to the parameter(s), the event instances of a particular type can be created. For example, the instigation event type can create different event instances by setting the parameter (e.g., the number of instigators) with different value.

When a scenario designer creates event instances, we also introduce the concept of scenario beat [8] as a higher-level construct to organize the events. In our design, each scenario beat is formally represented as a tuple: $b =< E, O, \delta >$, where E is the set of events contained in the beat, O is the set of ordering constraints on events in E, δ is the time interval in which all the events in the beat must be executed. The scenario beats are used to represent some typical situations in real-life, which involve the occurrence of single or multiple events within a certain period of time. All the scenario beats created by the scenario designer will be saved in the scenario beats repository and they serve as the key building blocks to construct a scenario. Each scenario is formed as a beat sequence, which contains a variable-sized, ordered set of beats, represented as: $S = (b_1, b_2, ..., b_m)$.

4.2 Data Training

In the *data training* module, a data-driven, simulation-based ANN training process is performed. The trained ANNs are used to estimate the MO intensities of a scenario. The MO intensity is used as a measurement to reflect the extent to which each defined MO is exercised in a given scenario. As each type of event is designed to exercise certain MO(s), the MO intensities of a scenario are related to the selection and arrangement of events in the scenario. For instance, a scenario with many instigation events should be attributed to a high MO intensity value for MO_2 (as defined in section 3). As contributed by all the events in a scenario, we use an aggregated MO intensity vector $\overrightarrow{I^{\text{a}}}$ to represent the total MO intensities of the scenario. Formally, $\overrightarrow{I^{\text{a}}} = [x_1^{\text{a}}, x_2^{\text{a}}, ..., x_n^{\text{a}}]$, where $\overrightarrow{I^{\text{a}}} \in \mathbb{R}^n_{\geq 0}$, x_i^{a} is the aggregated intensity of MO_i, n is the number of MOs in the scenario.

To train the ANNs for predicting $\overrightarrow{I^{\text{a}}}$, we first collect the player's performance data by using a set of sample scenarios. A random scenario generator, which randomly selects the beats from the scenario beats repository, is used for generating the sample scenarios. These sample scenarios are then sent to the simulator (i.e., our training game) for execution and the player performance data with respect to each MO are collected. To enable the large-scale data collection process, we design an AI player for playing the sample scenarios, instead of using real human players. The AI player can automatically issue commands in response to the events in a scenario and it imitates how human players (i.e., trainees) behave when playing scenarios. The use of the AI player is critical in our design, as we need to have a large set of sample scenarios for data training to ensure the prediction accuracy and it is generally infeasible to let human play testers to do this. The details of AI player design will be described in next section.

The AI player performance data collected at the end of the simulation are used to map to the MO intensities of the scenario. For each MO, we have a corresponding MO performance to measure the player's performance with respect to the MO. For instance, the MO of crowd anger control (i.e., MO_1) is measured by the number of angry agents being pacified by player over the number of all angry agents. The MO performances are used to reflect the MO intensities of a scenario for a given player. Each MO performance is (inversely) correlated to MO intensity. For instance, a low value of the number of angry agents being pacified over the number of all angry agents indicates a high intensity value for MO_1 of a scenario for the considered player. Currently, we assume a linear mapping between the MO performance to the corresponding MO intensity (Note that other type of mapping can be adopted to capture the relationship between the two). The MO performance is first normalized using min-max normalization and scaled to the value range of the MO intensity.

Based on the data collected from the simulation, we apply supervised learning to train the ANNs for estimating the MO intensities of scenarios. Each ANN is used to predict the MO intensity for a particular MO (i.e., each ANN has one output neuron) and the target outputs for training are the MO intensities mapped from the player's MO performances collected from the simulation. The input data for each ANN are the features extracted from the sample scenarios. In the current work, we use the following set of features as the descriptors to represent a scenario:

- Event-intensity features: these include number of instigators, number of agents turning angry, number of rumors being spread, number of jumping queue incidences, and number of food steal attempts in a scenario.
- Occupancy feature: the time period occupied by all the scenario beats (i.e., $\sum_{i=1}^{m} \delta_i$, m is the total number of beats in a scenario) over the total simulation time.
- Event-type feature: number of the distinct event types in a scenario.

To extract the values of the above scenario features for all sample scenarios, a *feature extractor* is used. The extracted feature values are fed as the inputs to the ANNs for data training. Apart from the scenario features, a set of player features (as described in the next section), which are used to reflect player's behaviors during the gameplay, is also fed as the inputs to the ANNs. Given all the inputs of the ANN and the

target outputs, each ANN is trained using back-propagation algorithm.

4.3 Scenario Generation

After the data training is completed, the *scenario generation* module is used to generate the scenarios based on trainer's inputs. In our design, a trainer can specify two key inputs for scenario generation. The first input is a mission objective MO intensity vector $\overrightarrow{I^{\text{in}}} = [x_1, x_2, ..., x_n]$, where $\overrightarrow{I^{\text{in}}} \in \mathbb{R}^n_{\geq 0}$, n is the number of MOs. By specifying the values in $\overrightarrow{I^{\text{in}}}$, the trainer can implicitly control the types of event instances to be generated, which are related to the training of different MOs. The second input is a skill level vector $\overrightarrow{L^{\text{in}}} = [l_1, l_2, ..., l_n]$, where $\overrightarrow{L^{\text{in}}} \in \mathbb{N}^n$ and it represents the trainee's estimated skill level with respect to each MO. The specification of $\overrightarrow{L^{\text{in}}}$ can be based on *a priori* knowledge about the trainee or a player model built based on the trainee's previous performance data. When multiple training sessions are conducted, $\overrightarrow{L^{\text{in}}}$ of a trainee can be updated iteratively from one session to another.

Based on the two trainer inputs, the desired MO intensity vector $\overrightarrow{I^{\text{d}}}$ is derived with the combination of $\overrightarrow{I^{\text{in}}}$ and $\overrightarrow{L^{\text{in}}}$. Each element x_i^{d} in $\overrightarrow{I^{\text{d}}}$ is proportional to the product of the corresponding element in $\overrightarrow{I^{\text{in}}}$ and $\overrightarrow{L^{\text{in}}}$, that is, $x_i^{\text{d}} \propto x_i l_i$. The purpose of doing this is to compensate the difference in trainees' skill levels. Pedagogically, a trainee should be assigned a scenario, which is appropriate to her/his existing skill levels. Thus, given the same MO intensities specified by the trainer, the desired intensities are adjusted according to different trainees' skill levels.

Once $\overrightarrow{I^{\text{d}}}$ is derived, the GA-based scenario generator is responsible to search for scenarios that can best match $\overrightarrow{I^{\text{d}}}$. The generation process starts from a randomly generated population of candidate scenarios, iteratively alters the events in the scenarios (via genetic operators), and evaluates the scenarios based on a given fitness function. In our design, the fitness function is defined as the Manhattan distance between $\overrightarrow{I^{\text{d}}}$ and the aggregated MO intensity vector $\overrightarrow{I^{\text{a}}}$ as:

$$f_{\text{fitness}} = d(\overrightarrow{I^{\text{d}}}, \overrightarrow{I^{\text{a}}}) \quad (1)$$

To obtain the values of $\overrightarrow{I^{\text{a}}}$ for a candidate scenario, the trained ANNs from the *data training* module are used. The feature values of the candidate scenario are used as the inputs for the ANNs, which produce the outputs that predict the aggregated MO intensities of the scenario (i.e., the elements in $\overrightarrow{I^{\text{a}}}$). The generation process terminates, when the best fitness in the population does not improve substantially for 20 iterations. The generated scenarios are then sent to the simulator for trainees to exercise.

5 AI Player Modeling

To automate the data collection procedure in our scenario generation framework, one important requirement is the use of AI player. In our training system, a player (either real or AI) can act as a commander and issue different types of commands (i.e., pacify, threaten, and arrest) to soldiers in the simulation. The soldiers will then perform the actions according to the issued command. Depending on the area where the events have occurred, the player has to switch from one area to another area in the simulation. Figure 3 shows a case when the AI player switches to the distribution area (one of the four areas as described in section 3) and controls the soldier agent to perform a specific action.

Figure 3: AI player-controlled soldier performing *pacify* action in distribution area.

Our design of the AI player intends to imitate how a real player behaves in our training game.

Specifically, we model a real player's personal characteristics with respect to attention, playing style and skill level. In our AI player model, the AI player's behaviors are characterized by a tuple of three key state variables: $< s, c, t >$, where s (switch time) is the time period during which the AI player stays in the current area until switching to the next area, c (issued command) is the command to be issued next, and t (action time) is the time to issue the next command.

The first variable s is used to reflect a player's attention characteristics. This means that during a period of time (i.e., s), a player can only focus and perform actions at a specific area of the simulation. After s has elapsed, the AI player will switch to the next area. The switch time s is computed as:

$$s = \max(T_a, s^{\mathrm{u}}) + A_{\mathrm{s}}\varepsilon_{\mathrm{s}}, \tag{2}$$

where T_a is the time period for AI player to complete all the actions within the current area a, s^{u} is a user-defined parameter to represent the average switch time of the AI player, ε_{s} is a random variable drawn from a given distribution of stochastic shift, with the mean of zero, and A_{s} denotes the strength of such stochastic influence. The first term in equation 2 implies that the AI player will switch to another area if there is no more action needed to perform in the current area or the time has exceeded the average switch time of the AI player. In our model, $T_a = t_f + \Delta t_{idle} - t_s$, where t_s is the start time when the AI player switches to the current area, t_f is the time when the AI player finishes all the actions in the current area, and Δt_{idle} is the idle time threshold value before changing area.

The second variable, issued command, determines the type of command to choose, given that the situation (e.g., the presence of both angry agents and instigators at the same time) requires the AI player to choose from a set of alternative commands. Given either two types of commands are applicable to choose at a certain point of time, the AI player has to choose the type of command based on its preference. In practice, such preference reflects the playing style of a player. For example, some people may prefer to *arrest* instigators rather than to *threaten* the stealing agents, while others may prefer the opposite. In our model, we use a probability ratio $p(c_{T_1}|c_{T_2})$ to describe the probability that the AI player chooses the T_1 type command over the T_2 type command. Here, we model the pair-wise preference. The selection probabilities of $p(c_{\text{arrest}}|c_{\text{threaten}})$, $p(c_{\text{threaten}}|c_{\text{pacify}})$ and $p(c_{\text{arrest}}|c_{\text{pacify}})$ are thus defined in our current design. In the case that all three types of commands can be chosen at a particular time, $p(c_{\text{arrest}}|c_{\text{threaten}})$ is used as these two types of commands are considered to be more crucial for controlling the situation. The issued command c, given a selection probability $p(c_{T_1}|c_{T_2})$ is determined as:

$$c = \begin{cases} c_{T_1}, \text{if } p(c_{T_1}|c_{T_2}) \geq R \\ c_{T_2}, \text{otherwise} \end{cases} \tag{3}$$

where R is a random variable drawn from a standard uniform distribution.

The last variable, action time, controls the time when the AI player issues a specific action. For the command of type T_i, the action time t_i is determined as:

$$t_i = \begin{cases} t_c + t_i^{\mathrm{r}}, \text{if } \mathcal{F}_i(t) = true \\ null, \text{otherwise} \end{cases} \tag{4}$$

where $\mathcal{F}_i(t)$ is the assertion function to determine when the AI player decides to issue the command c_{T_i}, t_i^{r} represents the response time that elapses from the time (i.e., t_c) the player *should* issue a command c_{T_i} to the time the player actually issues the command. t_i^{r} is used to reflect the player's skill level (i.e., how fast the player can react to issue the command of type T_i).

To decide when a command c_{T_i} needs to be issued, $\mathcal{F}_i(t)$ can be realized differently depending on the command type. For instance, for the *pacify* command, we define the following assertion function as:

$$\mathcal{F}_{\text{pacify}}(t) = \Theta[G(t, c) - \tau_{\mathrm{p}} - A_{\mathrm{p}}\varepsilon_{\mathrm{p}}(t)] \tag{5}$$

where $G(t, c)$ measures the number of angry agents at the time t in the current area c, $\Theta[x]$ is the Heavyside function which is one if $x \geq 0$ and zero otherwise, τ_{p} is a threshold value that specifies the number of angry agents for triggering the *pacify* action, $\varepsilon_{\mathrm{p}}(t)$ adds a stochastic shift to the threshold value, and A_{p} denotes the strength of such stochastic influence.

Table 2: Best MLP topology and its performance using testing set.

	MO$_1$	**MO$_2$**	**MO$_3$**
Best MLP Topology	24-41-1	24-29-8-1	24-54-27-1
Best Learning Rate	0.2	0.2	0.15
Average RMSE (std)	0.068 (0.006)	0.087 (0.014)	0.065 (0.005)

By adopting the AI player model as described above, we aim to generate human-like behavior to reflect how a real player actually plays our training game. Noteworthy, different human players may exhibit different levels of proficiency and playing styles when playing the game. To reflect the player's difference, we make the set of parameters $\{s^{\mathrm{u}}, p(c_{T_1}|c_{T_2}), t_i^{\mathrm{r}}\}$ as tunable parameters in our model. In our data collection, these parameters can be varied to represent different AI players and they are used as the player features as the part of the inputs of the ANNs. The value of each parameter is varied to cover the range from low to high values, which we observed from the data collected from real players.

6 Experiment

We evaluate the proposed scenario generation framework in the testbed training game (as described in section 3). In the following subsections, we will first describe the experiment to validate the AI player. We will then present the procedure and results of the ANN-based data training. Lastly, we will show the comparative evaluation of heuristic-based and data-driven approach.

6.1 AI Player Validation

To validate that our AI player model can reflect the human player's behaviors, we conducted experiments to compare the MO performances of the AI player and real player for playing identical scenarios. 20 participants (11 males and 9 females) were recruited to play some sample scenarios. Their MO performances and game-play features (e.g., average switching time) were recorded. Each player's game-play data are mapped to a corresponding AI player by assigning the appropriate values for the tunable parameters of the AI player model. The obtained AI player was then used to play the same scenario and its MO performances were recorded. We compared the performance difference between the human player and the corresponding AI player.

For the three MOs defined in section 3, three MO performance measures are used: P_{MO1}=number of pacified civilians/number of total angry civilians, P_{MO2}=number of arrested instigators/number of total instigators, and P_{MO3}=number of steals being stopped/number of total steal attempts. The value range of these three MO performance measure is $[0, 1]$. From our experiments, the average differences for P_{MO1}, P_{MO2} and P_{MO3} between the AI players and human players are: 0.028, 0.032 and 0.029 respectively. The two-sample two-tailed t-test was also conducted to check whether the AI player's performance is significantly different from that of the human player. The results yield the p-value of 0.68, 0.91, 0.86 for P_{MO1}, P_{MO2}, P_{MO3} respectively. This shows that the AI player's behaviors are generally consistent with human player's in terms of the MO performance.

6.2 ANN Data Training

To train the ANNs for scenario generation, a data collection process was conducted. 100 random scenarios were generated and each random scenario was played by a set of 10 different AI players (by varying the tunable parameters of the AI player model based on the sets of parameters of real players). The AI player MO performance data were recorded and a dataset of 1000 data points (i.e., 100×10) was collected. The three-way data split was performed on the collected dataset, with 20% of data were used as the testing set. To train and validate the ANNs, 8-fold cross-validation was used.

Each ANN is implemented using Multi-Layer

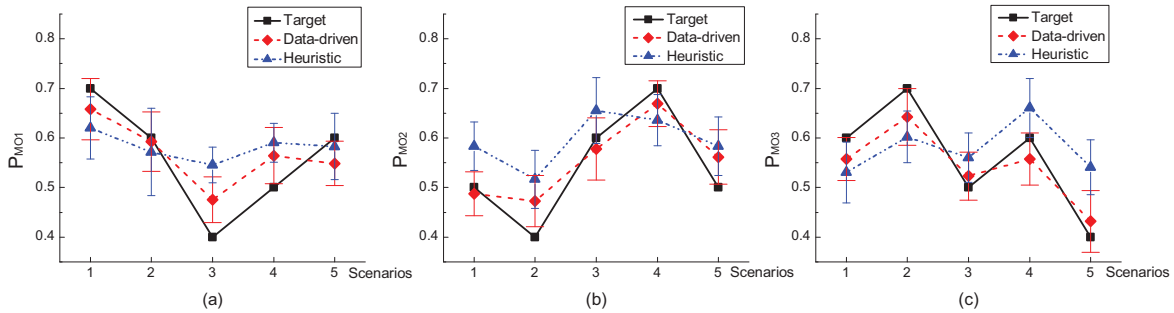

Figure 4: Target and actual MO performance of (a) P_{MO1}, (b)P_{MO2}, and (c) P_{MO3}.

Perceptron (MLP), with its inputs as the scenario and player features and its output as the predicted MO intensity. The targeted outputs are the MO intensities mapped from the MO performance obtained from the collected data. The inputs and output of MLP are normalized into [0,1] interval using min-max normalization. For model selection, MLP topologies of 2 hidden layers, with up to 80 and 30 hidden neurons, respectively, were investigated. The smallest MLPs that achieve the best cross-validation performance were selected.

Table 2 shows the performance of best MLP topologies (under best learning rate), based on the evaluation using testing set. The performance is measured by the root mean squared error (RMSE) between the targeted MO intensity and the predicted MO intensity using the ANNs. The results in Table 2 show the average value and standard deviation (in brackets) of 20 runs of data training. It can be seen that, despite the variation of the best MLP topology for different MOs, the ANNs can produce reasonable prediction results to match with the targeted values.

6.3 Heuristic-based vs. Data-driven Approach

To evaluate the effectiveness of the data-driven approach, we compare it with the heuristic-based approach we introduced in our previous work [8]. The heuristic-based approach utilizes a heuristic function to estimate the aggregated MO intensities of a given scenario based on the MO intensities of individual beats in the scenario. The MO intensities of scenario beats are pre-defined based on the designer's experience. In contrast, the data-driven approach uses the trained ANNs to do the MO intensity estimation. The evaluation was conducted by comparing the scenarios generated by the two approaches.

To generate a scenario, we first specified a target performance for each MO. The target performance describes the desired MO performance to be achieved when a trainee plays the scenario. Each target performance was then mapped into a MO intensity value. Here, we use the same mapping as in the ANN data training to transfer MO performance to MO intensity. The mapped MO intensities were further scaled by the estimated trainee' skill levels to derive the desired MO intensities (i.e., $\overrightarrow{I^{\mathrm{d}}}$ as in section 4.3). Given the $\overrightarrow{I^{\mathrm{d}}}$ as the inputs, we generated scenarios using each approach respectively.

To evaluate a generated scenario, we examined the actual MO performance obtained from the simulation against the target performance being specified. The smaller difference between the two indicates that the corresponding scenario matches closer to the desired MO intensities. Thus, we compare the two approaches by checking which approach can generate the scenario that yields smaller difference between the actual and target performance.

In our experiment, we generated a set of 5 scenarios using each approach with the specified target performance, as shown in Figure 4. To obtain the actual MO performance, we used the same AI player to play the scenarios generated by the two approaches. For the three types of MO performance as described in section 6.1, the target performance and actual performance of the scenarios generated by heuristic-based and data-driven approach are shown in Figure 4.

Note that for each scenario, we generated 10 instances of the scenario using the same target performance. Each actual performance shown in Figure 4 is the average value of ten scenario instances.

Based on the results shown in Figure 4, we calculated the average difference between the target and actual performance across all 50 scenarios (i.e., 5 sample scenarios $\times$ 10 instances for each sample scenario). The average differences for data-driven approach are 0.048, 0.040, and 0.039 for p_{MO1}, p_{MO2}, and p_{MO3} respectively; whereas the average differences for heuristic-based approach are 0.072, 0.081, and 0.085 respectively. The two-sample one-tailed t-test (with sample size = 50) was also conducted to check whether the difference between the target and actual performance using data-driven approach is lower than the one using heuristic-based approach. The results yield the p-value of $p < 0.001$, $p < 0.01$, $p < 0.001$ for P_{MO1}, P_{MO2}, P_{MO3} respectively. This shows that our data-driven approach outperforms the heuristic-based approach, as it is capable to generate the scenarios that match closer to the specified target performance.

7 Conclusion

This paper presents our first step in designing a data-driven scenario generation framework for serious games. In the proposed framework, a data training process is introduced to train the neural networks based on simulation data for predicting the scenario's intensities with respect to different mission objectives. To facilitate the data collection from simulation, an AI player model is also designed to imitate the behaviors when human player plays scenarios. While our experiments show the ANN-based prediction achieves reasonable accuracy, we will continue to improve our model's performance by considering more scenario features in our data training process. Furthermore, we will conduct experiments involving human players to demonstrate the training effectiveness of our system.

References

[1] R. Hill, J. Gratch, S. Marsella, J. Rickel, W. Swartout, and D. Traum. Virtual humans in the mission rehearsal exercise system. *Künstliche Intelligenz*, 4(03):5–10, 2003.

[2] B. Magerko, B.S. Stensrud, and L.S. Holt. Bringing the schoolhouse inside the box-a tool for engaging, individualized training. In *Proceedings of the 25th Army Science Conference*, 2006.

[3] M.O. Riedl, A. Stern, D. Dini, and J. Alderman. Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications*, 4(2):23–42, 2008.

[4] W.L. Johnson, J.W. Rickel, and J. Lester. Animated pedagogical agents: Face-to-face interaction in interactive learning environments. *International Journal of Artificial Intelligence in Education*, 11(1):47–78, 2000.

[5] A. Meluso, M. Zheng, H.A. Spires, and J. Lester. Enhancing 5th graders' science content knowledge and self-efficacy through game-based learning. *Computers & Education*, 59(2):497–504, 2012.

[6] A. Zook, S. Lee-Urban, M.O. Riedl, H.K. Holden, R.A. Sottilare, and K.W. Brawner. Automated scenario generation: Toward tailored and optimized military training in virtual environments. In *Proceedings of the 7th International Conference on the Foundations of Digital Games (FDG)*, pages 164–171, 2012.

[7] L. Luo, H. Yin, W. Cai, M. Lees, and S. Zhou. Interactive scenario generation for mission-based virtual training. *Computer Animation and Virtual Worlds*, 24(3-4):345–354, 2013.

[8] L. Luo, H. Yin, J. Zhong, W. Cai, M. Lees, and S. Zhou. Mission-based scenario modeling and generation for virtual training.

In *Proceedings of the 9th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2013.

[9] B.F. Allen, N. Magnenat-Thalmann, and D. Thalmann. Politeness improves interactivity in dense crowds. *Computer Animation and Virtual Worlds*, 23(6):569–578, 2012.

[10] L. Sun, A. Shoulson, P. Huang, N. Nelson, W. Qin, A. Nenkova, and N.I. Badler. Animating synthetic dyadic conversations with variations based on context and agent attributes. *Computer Animation and Virtual Worlds*, 23(1):17–32, 2012.

[11] N. Shaker, G.N. Yannakakis, and J. Togelius. Towards automatic personalized content generation for platform games. In *Proceedings of the 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2010.

[12] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelback, and G.N. Yannakakis. Multiobjective exploration of the starcraft map space. In *Proceedings of the 2010 IEEE Symposium on Computational Intelligence and Games*, pages 265–272, 2010.

[13] J. Porteous, M. Cavazza, and F. Charles. Narrative generation through characters' point of view. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1297–1304, 2010.

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

[15] K. Hullett and M. Mateas. Scenario generation for emergency rescue training games. In *FDG*, pages 99–106, 2009.

[16] G.A. Martin, C.E. Hughes, S. Schatz, and D. Nicholson. The use of functional L-systems for scenario generation in serious games. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 2010.

[17] G.N. Yannakakis, H.P. Martínez, and A. Jhala. Towards affective camera control in games. *User Modeling and User-Adapted Interaction*, 20(4):313–340, 2010.