International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Framework for Emotion based Adaptive Game State Selection Method using Multivariate Normal Distribution

[1]Sreenarayanan N. M., [2]Dr. Partheeban N.

Submitted: 04/02/2024 **Revised**: 13/03/2024 **Accepted**: 21/03/2024

Abstract: An adaptive game design is an attractive area for the researchers to participate and contribute more by including various emotional factors, which not only includes emotional factors for a player. The player emotions are directly affect game success factors and the emotions can be measured directly through various facial, speech, and text expressions. This can be indirectly calculated through efficiency of a player by calculating success factor and time to complete each state of a game. In this paper, we have proposed an adaptive game state selection method based on multivariate normal distribution. The proposed method uses two important factor for deciding next state selection from the current level of a game, time to complete one single state and complexity of states within the particular level. The proposed method is a kind of slow-learning technique using multivariate normal distribution method. The experiment evaluation is done by using three different game strategy with 150, 280, and 324 iterations. We have used two other distribution functions for taking accuracy and average error ratio for the proposed method. The performance evaluation shows that the proposed method achieves 79.3 % accuracy and 20.7 % average error ratio. The exponential and poison distribution achieves accuracy of 73.7 % and 72.3 % respectively.

Keyword*: Adaptive Game Design, Emotional identification, Complex game environment, Normal distribution, Poison distribution, Exponential distribution, Multivariate Dataset*

1. Introduction

Artificial Intelligence provides an excellent solution for real-time problems with optimal solution and this field has been achieved incredibly over the past two decades. Emotion based game tree design is emerging research area and AI based optimization techniques are providing an efficient solution for generating well suited game scenario. The primary characteristic of a video game is to attracts and motivate the players to participate actively in the game. Any successful video game has the following standard features as common elements, excellent goals, active states, and interactive graphical user interface. The game designer has to consider these basic elements during the game design.

The major area of game design has changed tremendous way and they have consider each and every movement of player participation. Nowadays, games are designed from the player point of view and more weightages are given for player emotional values. The technologies involved in the game design continuously evolved and these games are adaptively changes their states based on the ability of player 2] 3]. Maintaining static level complicity for designing a game tree is no longer suitable for current scenario. The modern game world satisfies the player expectations and adaptabilities quickly by recognizing user needs and emotions through various probabilistic models.

Research Gaps

The traditional game design has the following unsolved issue,

1. The traditional game design does not support for adaptive state selection
2. The existing game design uses static model for planning a game tree and these techniques are not supporting for dynamic state selection based on user emotions
3. Most of the game tree designs are not taking user emotions as primary factor for selecting the next state in the running mode of game
4. The traditional methods are constructing the game trees based on static layouts and these techniques does not provide achieves user satisfaction

2. Related Work

Difficulty adjustment by dynamically is used to change the mode of game either easy or hard. The dynamic adjustment achieves mode changing is an effective way. Xue et al. [4] presented a DDA framework based on global optimization to create more participation into the entire game and they have modelled a players progression as a probabilistic graph. Segundo et al. [5] proposed a parameter manipulation technique for

[1]*Research Scholar,* [2]*Professor*
[1,2]*School of Computing Science and Engineering*
[1,2]*Galgotias University, Greater Noida, Uttar Pradesh, India*
[1]*sree.narayanan1@gmail.com,* [2]*n.partheeban@galgotiasuniversity.edu.in*

creating difficulty level dynamically and aims to make an improvement in the gaming experience. It is necessary to emphasize that the proposed approach uses probabilistic calculations that will be used in the challenge function. A questionnaire was applied to a sample of students in order to determine whether there were statistically significant differences in the perception of game play, difficulty of the game and desire to play several times with and without the use of the technique. Bunian et al. [7] modelled an approach using data collected from players view point like role-playing game (RPG). They developed an approach with hidden Markov model for the behavor of a player with individual differences and generate behavioural feature extraction for classifying player characteristics [6]. Khajah et al. [8] designed a gamming method using Bayesian optimization techniques and this increase the player participation in the game. Engagement in a game for several minutes is measured by players inside the game mode, projections of how long other players will be in game, and to conduct a post-game survey.

Pedersen et al. [10] examines the relationship with design level parameters as platform games, individual playing characteristics and experience of a player. These investigation parameters are closely related to measuring the individual players attitude, which includes, various emotional feelings. Yannakakis et al. [11] has taken an attempt to construct metric models for designing "Bug Suasher" game in a Playware environment. They have used a set of numerical features collected through interaction process with children's. Sequential forward selection technique and artificial neural network bas n-best feature selection algorithms are used to build function and feature sets for creating fun for this game. Shaker et al. [12] design a model based on predicted player experience by using features of level design and style of game playing. These game models are constructed by using preference learning.

Cowley et al. [13] presented a novel approach to design an expert domain knowledge by using theoretical framework based on behavior and game design patterns. They have developed a model known as "Behavkets" by using player behavior with respect to psychological theory. They have presented a theoretical supporting of psychology, player modeling, temperament theory, and game composition.

Pedersen et al. [14] uses computational intelligence methods to develop a quantitative approach of player experience for a platform game. Dynamically construct the levels for a 2D games with continues suitable challenges by using level generation and machine learning [15]. Carvalho et al. [16] proposed an approach for automatic game level design using computational model based on player environment and generative system. This generative system signifies with the combination of constraint satisfaction and genetic algorithms technique. They have constructed a fitness function for creating a fun levels for different games. Spronck et al. [17] presented a novel approach known as "dynamic scripting" to meet the player requirements. In this method, an adaptive rule-base is used for creating an intelligent opponents on the fly.

Ultima online is one of the popular online role-playing game and this provides an interaction between all the involved players [18]. Togelius et al. [19] discussed about the issues related to automatically constructing tracks tailor made to enhance the enjoyment of individual player. This method uses evolutionary technique for constructing the tracks in the racing game. A method has been proposed to evaluate effects of video game playing on motor learning method and their potential to increase patient involvement with therapy related treatments [20]. Recently many research papers reviews and analyze the adaptive game designing for collaborative and educational system [21][22]. In the concept of educational applications, different studies are linked game objects to motivate student as a player and learner types [23][24][25][26]. The main idea of adaptive game design is use to make an literature analysis in the field of education to enhance the learners level of understanding, to identify a suitable adaptation method. This technique creates an impact on continuous improvement through performance to motivation in the learning environment

Contributions

This paper proposes following contributions in the adaptive game state selection methods,

1. This method uses multivariate normal distribution function for selecting suitable or sustainable state selection from the given or available states in a particular level
2. We have Time complexity and state complexity as two parameters for the selection process. These two parameters are evenly distributed over all the stage in the game to select an appropriate state
3. The experiment evaluation conducted for the proposed method by using poison distribution and exponential distribution
4. We have created a gaming dataset with seven attributes for conducting the experiment evaluation

The reaming section paper is organized as follows, section 2 discussed about the related paper published in adaptive game designing. Section 3 discussed about the proposed adaptive game design model using multivariate normal distribution function. The performance

evaluation for the proposed method given in the section 4 and section 5 conclude evaluation and future directions for the proposed method.

3. Proposed Adaptive Game Design Model

Definition of Game Tree Model

The game tree design consists of seven fundamental components and these components are the basic elements to construct the game structure. This section discussed about the elements in game tree construction.

a. Internal States (IS) This describe about each state of a game and this will be an finite set of states. If G is a game then it has set of internal states defined as $X \leftarrow (x_i, 0 \leq i \leq M)$, here M is defined as maximum number of internal states in a game tree

b. Start State The initial starting state of a game tree denoted as $x_0 \in X$

c. Terminate State This will be a collection of states from $x_j \in X, 1 \leq j \leq M_{end}$ and $M_{end} < M$. The ending state might consist of either any one of the following state Win, Loss, or Tie. The non-empty terminate states are the sub-set from internal states, $T \subseteq X$

d. Status Describe about the current state information of a game T, which may be either ongoing or terminate status

e. State Transaction A transaction function represents a transition between current sate to next state by applying specific set of inputs, this can be defined as follows,

a. The function of a single state transition defined as $f: X \rightarrow 2^X$

b. $f(x)$ consist of set of successor states$x_i \in X$ buddy medium

f. Complexity level of a Current State This will provide a label associate with a current state and complexity level from $\{UltraHard, Hard, Medium, Easy, Buddy\}$. The internal state $x_i \in X$ labeled with,

$$C_i \leftarrow \{UltraHard, Hard, Medium, Easy, Buddy\}$$

g. State Value Each internal state assigned with a real-value $V_i$to each $x_i \in X$ within the interval of $[-1t\ to\ 1]$.

Each internal state $x_i \in X$ is defined with four tuples $\{x_i, C_i, V_i, AvgT_i\}$. The $AvgT_{x_i}$denotes the average time to complete an internal state x_i (equation 10). The figure 1 shows the general structure of a game tree with based nodes.

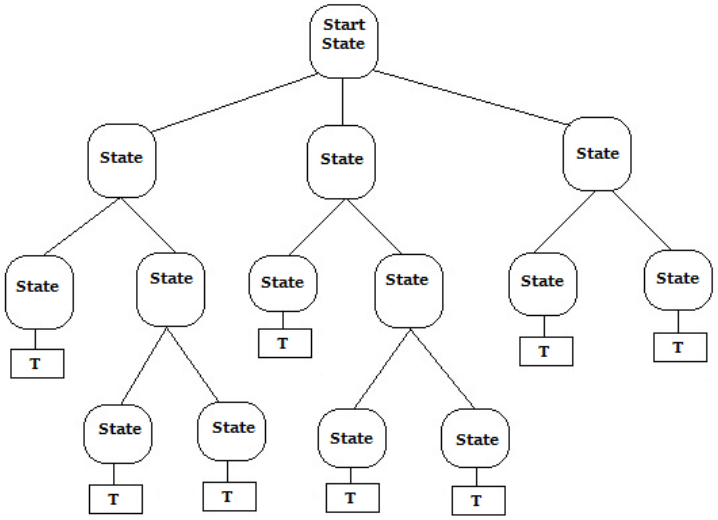

Figure 1: General Game Tree Construction

In game deign, game tree construction taking an important role and real time game trees are denoted as behavior tree. The following section discussed about the fundamental concept of behavior tree.

Fundamental Concepts of Game Trees

A Behavior Tree has been designed as a directed graph with standard nodes of *root*, *child*, *parent*, and *leaf* nodes. All the leaf and non-leaf nodes are represented as *running nodes* and *condition flow nodes*.

The Behavior Trees are associated with an AI entity and this will be executed based on timely manner with respect to the input.

The execution starts from the root node in BT and this generates *Ticks* signals based on a given incidence. The *Ticks* signal activates the node through execution and this will be move forward to one or more children as a ticked node.

Child nodes are executed if it receives *Ticks signal*. The child node has the responsibility to answers to the parent node by using a return signal statement "*Running"*, if the

goal attained then it will returns "*Success"*, otherwise it will return "*Failure"*.

If the child node could not completed the required task then it needs more AI steps to complete the task, then it will returns "Running" [17].

The "*Error*" statement is use to indicate the an error occurred in a processing node. These errors may be a programming error [15].

The traditional formulation of Behavior Tree has three main categories of control flow nodes with two categories of executions nodes. The following section explain in detail about the control flow nodes

a. Sequence Flow Control

The sequence is a set of actions are completed and few more conditions checks needs to performed in a sequence order, and the success of one action has manipulates the action next node. The sequence nodes are guided by the "*Ticks*" to other children nodes from left to right until reach to find the child node which returns either "*Failure*" or "*Success*". The following algorithm 1is explains about the role of sequence flow control to access a child node from the N children. If all its children nodes return signal of "*Success*" then the child node will returns a signal of "Success". If any child node returns "Running" status then the control will be transfer to next movement in the game, whereas in the case of "Failure" status, the action of a player will be completely changed or completed cancel. The sequence node in a Behavior Tree is indicated with "→".

b. Fallbacks Flow Control

Fallbacks [2] are defined with alternative set of actions to achieving a similar goal. The following algorithm 2 explains about the Fallback steps in detail. The Fallback node directs the "*Ticks*" to the corresponding to left side children till it find a child node that returns any one of the signal like "*Success*" or a "*Running*". If all the children nodes are returning "*Failure*" then it will return "*Failure*". The Fallback algorithm will not send the "*Ticks*" to next level if a child node returns "*Running*" or "*Success*". The "?" is used to indicate the Fallback node in a Behavior Tree

c. Parallel Nodes

The Parallel node performs Tick process in simultaneously. The following algorithm 3 shows that form the M child nodes out of N children returns the value of *Success*, then the node will be a parallel node. If $N - M$ returns Failure thus translation success unbearable, then it returns a signal of Failure

d. Reference Components

For referring another BT from an old BT we used a *reference* component and this is use to connect simple sub behaviors. The new BT can be constructed and incorporate with the existing BT by using reference component. The reference component status code are produced by the existing BT to make link with other BT. A double-boxed labelled nodes are used to represent a reference component with an identifier as a name of another BT

e. Action Component

The action component updates the entity state by using an action, which agrees to execute the exact game code. For example the action in the game is defined by including movement in the simulated world, altering internal state, sound playing or applying some special logics. If the action accomplished successfully then it will return "Success". If the action could not be completed means it will return "Failure". In the case of returning the Running status, the action to be continued with new set of actions

A condition component is a Boolean function and this function will return either true or false. These functions can be tested with direct questions. If the Boolean function returns Success then the status code is true, otherwise it returns false. The condition component is represented with question mark in a labelled box

The selector component control flow is used to take a choice. The selector processes all the child nodes from the direction of left side to right side and this will returns a signal as *Success* mode if any one of the child node returns a signal of achievement as Success. It executes a signal as *Running* code if any one of its children node returns signal as a *Running* code. If the child node returns signal value as *Failure*, then same process will be continued by the selector for the following component. This selector node is indicated in a labelled with question mark as circle.

Algorithm 1: Sequence

1. **For** Each child from BT **do**
2. $StatusChild(Node_i) \leftarrow Tick\big(Child(Node_i)\big)$
3. $\boldsymbol{Check\ condition}\ (StatusChild(Node_i) = \text{“}Running\text{”})$
4. $return$ "Running"
5. $\boldsymbol{Else\ condition}\ (StatusChild(Node_i) = \text{“}Failure\text{”})$
6. $return\ "Failure"$
7. $\boldsymbol{End\ For}$

8. $return\ "Success"$

Algorithm 2: Fallback

1. **For** Each child from BT **do**
2. $StatusChild(Node_i) \leftarrow Tick\big(Child(Node_i)\big)$
3. $\boldsymbol{Check\ condition}\ (StatusChild(Node_i) = Running)\ \boldsymbol{Then}$
4. $return$ "Running"
5. $\boldsymbol{Else\ condition}\ (StatusChild(Node_i) = Success)$
6. $return\ "Success"$
7. $\boldsymbol{End\ For}$
8. $return\ "Failure"$

Algorithm 3: Parallel

1. *N* denoted total Nodes in Behavior Tree
2. *M* denoted subset of Nodes in Behavior Tree
3. **For** Each child from BT **do**
4. $StatusChild(Node_i) \leftarrow Tick\big(Child(Node_i)\big)$
5. $\boldsymbol{If}\ (\forall M\ StatusChild(Node_i) = Success)\ \boldsymbol{Then}$
6. $return\ "Success"$
7. $\boldsymbol{Else\ If}\ (\forall (N-M)\ StatusChild(Node_i) = Failure)\ \boldsymbol{Then}$
8. $return\ "Failure"$
9. $\boldsymbol{End\ For}$
10. $return\ "Running"$

4. Proposed Active Game Stage Selection Method

The players are taking an active participant role in playing games and this has to be more interactive based on the current emotion and state of mind of a player. The players are having different style to play the games and it needs to be evaluated for designing a more interactive games. One of the primary goal of a game designer is to make the player to play the game continuously. This can be achieved through continuously watching the user behavior. In the proposed adaptive game stage selection method, we have introduced new factor for measuring the game stage complexity, through which we can decide how to assign a new stage in the current game environment.

Game Stage Complexity

An efficient Game design should attract players and crates more involvements to participate in the game activities. This is an optimization problem for the game designers and this has to provide an optimum solution for the growth of game stages. In the proposed game stage selection method, we have used two primary factors, time complexed to complete one single stage and complexity level of each stage. The following equation use to calculate time complexity and stage complexity,

$$CLevel_{CurrtS} \leftarrow \lambda + \sum_{i=1}^{N} T_{x_i}.C_{x_i} \quad \leftarrow (1)$$

$$\lambda_{(T_S,C_S)} \leftarrow \frac{1}{2\pi.\sigma_{T_S}.\sigma_{C_S}.\sqrt{1-\rho^2}} e^{-\frac{1}{2(1-\rho^2)}\left[\left(\frac{T_{x_i}-\mu\_T_S}{\sigma_{T_S}}\right)^2+\left(\frac{C_{x_i}-\mu\_C_S}{\sigma_{C_S}}\right)-2.\left(\frac{T_{x_i}-\mu_{T_S}}{\sigma_{T_S}}\right).\left(\frac{T_{x_i}-\mu\_C_S}{\sigma_{C_S}}\right)\right]} \leftarrow (2)$$

$$\mu_{T_S} \leftarrow \frac{\sum_{i=1}^{N} T_{x_i}}{N} \quad \leftarrow (3)$$

$$\sigma_{T_S} \leftarrow \sqrt{\frac{1}{N}\left(\sum_{i=1}^{N}(T_{x_i}-\mu_{T_S})\right)^2} \quad \leftarrow (4)$$

$$\mu_{C_S} \leftarrow \frac{\sum_{i=1}^{N} C_{x_i}}{N} \quad \leftarrow (5)$$

$$\sigma_{C_S} \leftarrow \sqrt{\frac{1}{N}\left(\sum_{i=1}^{N}(C_{x_i}-\mu_{C_S})\right)^2} \quad \leftarrow (6)$$

$$T_{NextS} \leftarrow \frac{T_{CurrtS}}{N} \quad \leftarrow (7)$$

Here, ρ values may be varying from [0,1]. The proposed game state selection algorithm illustrated in algorithm1 and algorithm 2 using Single and Multi-variate Normal Distribution. The algorithm 1 discussed about the

personalized game tree construction based on the user level of playing. The algorithm 2 is used to select the next suitable state from the current state of a game based on complexity values. The average time taken for completing a single state $x_i \in X$ with maximum iterations computed as follows,

$$AvgT_{x_i} \leftarrow \frac{\sum_{i=1}^{Max} T_{x_i}}{N} \qquad \leftarrow (8)$$

Algorithm 1: Game Next Stage Selection Input: Game Tree G (defined in Definition 1) Output: Game Wining Status and Terminate State $x_j \in T$	
1.	$CurrtState \leftarrow x_0$ // assign a initial state of a Game G
2.	Assign $T_{S_0} \leftarrow 0$ and $C_{S_0} \leftarrow 0$
3.	$While(CurrtState \notin T)$
4.	Begin
a.	Start $PLAY(CurrtState)$
b.	$T_{S_i} \leftarrow TimeTaken(CurrtState)$
c.	$CurrtState \leftarrow Next - State\ Selection\left(CurrtState, T_{S_i}, C_{S_i}\right)$
5.	End
6.	$\boldsymbol{Return}\ \left(Status(CurrtState)\right)$

Algorithm 2: Game Next State Selection Input: $CurrtState$, Time taken for completing previous state, Complexity of previous state Output: Next Optimal State $x_i \in X$	
1.	Compute μ_{T_S} and μ_{C_S} using equation (3) and (5)
2.	Compute σ_{T_S} and σ_{C_S} using equation (4) and (6)
3.	For each $x_i \in X$ do
a.	Compute average time $AvgT_{x_i}$ using equation (8)
b.	Compute complexity level of state x_i by using equation (1)
c.	Compute average time μ_{T_S} consumption rate using equation (3)
d.	$If\left[\left(\mu_{T_S} \leq AvgT_{x_i}\right) and\ \left(CLevel_{CurrtS} \leq C_{x_{i-1}}\right)\right] then\ \boldsymbol{Return}(\boldsymbol{x_i})$
4.	End For

5. Result and Discussion

Dataset

We have designed a dataset for game tree construction based on the model assumption defined in the definition part. In which the following parameters are added, game level (GL), level complexity (LC), State number $(S_{\#})$, individual state complexity $\left(ISC_{x_i}\right)$, average time for completing particular state $\left(AvgT_{x_i}\right)$, state nature(S_N), and winning possibility $\left(WP_{x_i}\right)$. The following table provides sample of data feed taken from dataset,

GL	$S_{\#}$	LC	ISC_{x_i}	$AvgT_{x_i}$ (seconds)	S_N	WP_{x_i}(%)
1	3	0.21	0.18	38	NT	93
1	6	0.21	0.16	33	NT	95
2	1	0.26	0.25	67	NT	91
2	5	0.26	0.27	64	NT	91.5
3	6	0.29	0.36	93	NT	89.4
3	2	0.29	0.35	96	NT	88.7
5	4	0.61	0.58	182	NT	63.4
7	6	0.72	0.67	192	NT	18.7
9	2	0.84	0.78	162	T	-1
16	4	0.89	0.85	183	T	1

Table 1: Sample Dataset from Game Tree construction with 16 Levels

We have used three different dataset with 16 levels, 24 levels, and 30 levels. The table 1 contains the samples taken from a game dataset, which contains 16 levels. The complexity of level in a game tree is assigned in as increasing order within the interval of $[0, 1]$. The 0.01 indicates complexity is very less and 0.93 indicates high

complexity. The average time for completing individual state indicates the average time taken for completing one single state. State nature attribute indicates the current state is either terminate state T or non-terminate state NT. The winning probability attribute contains a non-linear integer value as a percentage of success in the game. If the winning probability is 1 then Game completed with wining and if it is -1 means then the player Loss the game. If it is 0 then game ended with tie mode.

Experimental Evaluation

We have created three different dataset by varying the levels from 16, 24, and 30. These data sets are contains 553, 417, and 623 individual state entries respectively. We have used simple python code for writing the proposed adaptive game state selection method. The experiment evaluation conducted with different set of iterations for each game dataset. The experiments are coined with 150, 280, and 324 iterations for each game and we have measure two parameters, time take for completing all the iterations and average success, failure and tie in the game. The following equation (8), (9), and (10) used to calculate the average winning, losing, and tie for each game with different number of iterations $(\#_{Played})$. Here $\#_{Win}$, $\#_{Loss}$, and $\#_{Tie}$ indicates number of times success, loss, and tie in the game. The accuracy for the proposed is calculated by using the equation 11 and equation 12 is used measure the average error ratio.

$$Avg_{Success} \leftarrow \frac{\#_{Win}}{\#_{Played}} \quad \rightarrow (8)$$

$$Avg_{Loss} \leftarrow \frac{\#_{Loss}}{\#_{Played}} \quad \rightarrow (9)$$

$$Avg_{Tie} \leftarrow \frac{\#_{Tie}}{\#_{Played}} \quad \rightarrow (10)$$

$$Accuracy \leftarrow \frac{Avg_{Success} + Avg_{Tie}}{Avg_{Success} + Avg_{Loss} + Avg_{Tie}} \quad \rightarrow (11)$$

$$Avg_{ErrorRatio} \leftarrow \frac{Avg_{Loss}}{Avg_{Success} + Avg_{Loss} + Avg_{Tie}} \quad \rightarrow (12)$$

We have used multivariate normal distribution in the proposed method. In the experiment evaluation, we have used following two types of distribution mechanisms, exponential and poison distribution. The nature of these two distributions are different and equation (13) and (14) illustrate process of distribution,

$$\lambda^{EDis}_{(T_S,C_S)} \leftarrow (\mu_{T_S} + \mu_{C_S}).e^{-(\mu_{T_S}.T_{x_i} + \mu_{C_S}.C_{x_i})} \quad \rightarrow (13)$$

$$\lambda^{PDis}_{(T_S,C_S)} \leftarrow \frac{e^{-(\mu_{T_S}+\mu_{C_S})}.\mu_{T_S}^{T_{x_i}}.\mu_{C_S}^{C_{x_i}}}{(T_{x_i} + C_{x_i})!} \quad \rightarrow (14)$$

Proposed Method with different Distributions	Number of Iterations									Accuracy	Avg. Error Ratio
	150			280			324				
	Win	Loss	Tie	Win	Loss	Tie	Win	Loss	Tie		
Multivariate Normal Distribution (MND)	95	19	36	152	42	76	201	55	68	85.6	16.3
Poison Distribution (PD)	92	24	34	137	49	84	189	56	79	83.1	17.9
Exponential Distribution (ED)	86	34	30	129	57	84	188	60	76	81	19

Table 2: Performance Evaluation for Game Strategy 1 with 16 Levels

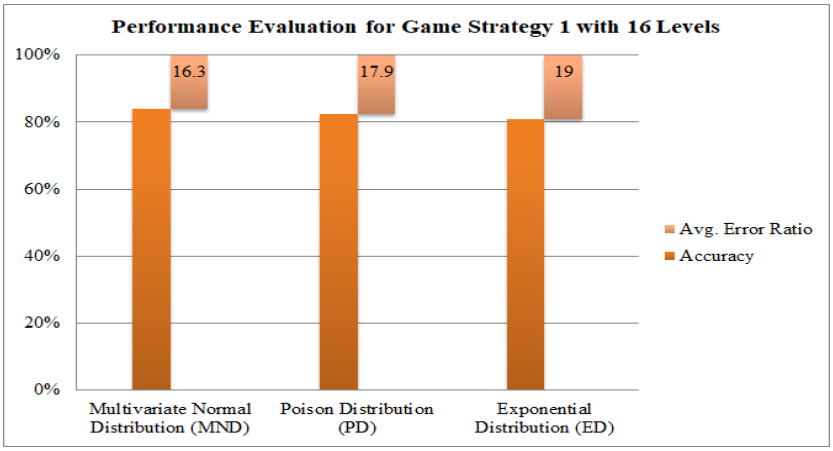

Figure 2: Performance Evaluation for Game Strategy 1 with 16 Levels

Proposed Method with different Distributions	Number of Iterations									Accuracy	Avg. Error Ratio
	150			280			324				
	Win	Loss	Tie	Win	Loss	Tie	Win	Loss	Tie		
Multivariate Normal Distribution (MND)	104	21	25	162	44	74	221	52	61	86.1	13.9
Poison Distribution (PD)	94	24	30	150	52	78	203	62	59	83.6	17.4
Exponential Distribution (ED)	85	31	34	151	49	80	192	71	61	79.3	20.7

Table 3: Performance Evaluation for Game Strategy 2 with 24 Levels

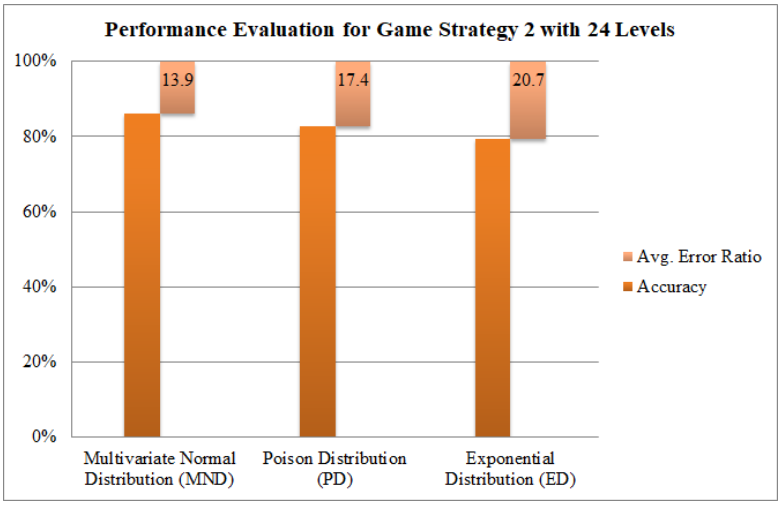

Figure 3: Performance Evaluation for Game Strategy 2 with 24 Levels

Proposed Method with different Distributions	Number of Iterations									Accuracy	Avg. Error Ratio
	150			280			324				
	Win	Loss	Tie	Win	Loss	Tie	Win	Loss	Tie		
Multivariate Normal Distribution (MND)	97	14	39	171	39	70	215	51	58	90.6	9.4
Poison Distribution (PD)	89	21	40	158	52	70	209	61	44	85.3	14.7
Exponential Distribution (ED)	82	27	41	163	48	69	205	58	61	84.4	15.6

Table 4: Performance Evaluation for Game Strategy 3 with 30 Levels

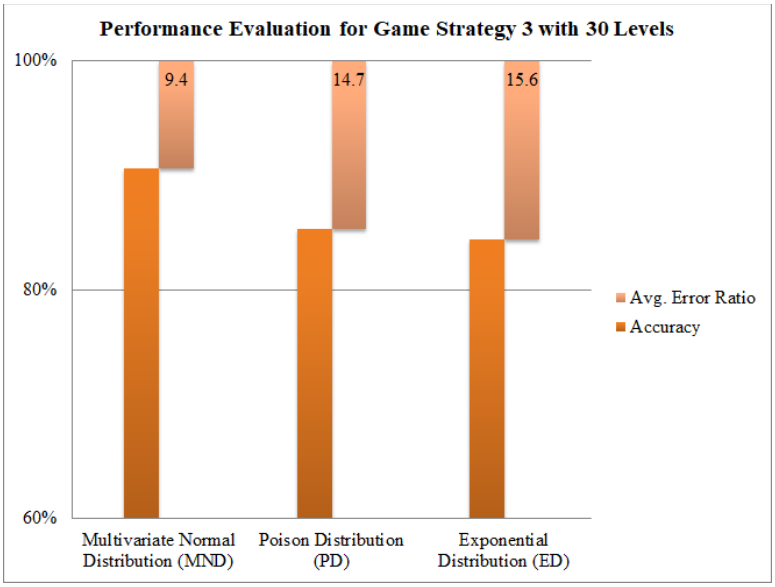

Figure 4: Performance Evaluation for Game Strategy 3 with 30 Levels

Performance Evaluation

This section discussed about the performance evaluation for the proposed adaptive game state selection method based on different set of iterations like 150, 280, and 324. The accuracy and average error ratio for the proposed method shows that 78.7% and 21.3% respectively. The poison and exponential distribution methods are providing accuracy of 73% and 72%. The proposed adaptive game design methods are performing well.

6. Conclusion

In this paper, we have proposed an adaptive game state selection method using multivariate normal distribution function for selecting game states in a linear way. The proposed method uses two important factors for deciding next state selection from the current level of a game, time to complete one single state and complexity of states within the particular level. The proposed method is a kind of slow-learning technique using multivariate normal distribution method. The experiment evaluation is done by using three different game strategy with 150, 280, and 324 iterations. We have used two other distribution functions for taking accuracy and average error ratio for the proposed method. The performance evaluation shows that the proposed method achieves 79.3 % accuracy and 20.7 % average error ratio. The exponential and poison distribution achieves accuracy of 73.7 % and 72.3 % respectively.

Reference

[1] S. Bunian, A. Canossa, R. Colvin, and M. S. El-Nasr, "Modeling individual differences in game behavior using HMM," in *Proceedings of the 13th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)*, 2017

[2] P. Sweetser and P. Wyeth, "GameFlow," Computers in Entertainment, vol. 3, no. 3, 2005.

[3] K. M. Gilleade, A. Dix, and J. Allanson, "Afective videogames and modes of afective gaming: Assist me, challenge me, emote me," in Proceedings of the 2nd International Conference on Digital Games Research Association: Changing Views: Worlds in Play (DiGRA '05), 20, 16 pages, Vancouver, Canada, June 2005

[4] S. Xue, M. Wu, J. Kolen, N. Aghdaie, and K. A. Zaman, "Dynamic Difculty Adjustment for Maximized Engagement in Digital Games," in Proceedings of the 26th International Conference, pp. 465–471, Perth, Australia, April 2017

[5] C. V. Segundo, K. Emerson, A. Calixto, and R. P. Gusmao, "Dynamic difculty adjustment through parameter manipulation for Space Shooter game," in Proceedings of SB Games, Brazil, 2016

[6] H. Hsieh, Generation of Adaptive Opponents for a Predator-Prey Game, Asia University, 2008.

[7] S. Bunian, A. Canossa, R. Colvin, and M. S. El-Nasr, "Modeling individual diferences in game behavior using HMM," in Proceedings of the 13th AAAI Conference on Artifcial Intelligence and Interactive Digital Entertainment (AIIDE-17), 2017.

[8] M. M. Khajah, B. D. Roads, R. V. Lindsey, Y.-E. Liu, and M. C. Mozer, "Designing engaging games using Bayesian optimization," in Proceedings of the 34th Annual Conference on Human Factors in Computing Systems, CHI 2016, pp. 5571–5582, San Jose, Calif, USA, May 2016.

[9] A. Hintze, R. S. Olson, and J. Lehman, "Orthogonally evolved AI to improve difculty adjustment in video games," in European Conference on the Applications of Evolutionary Computation, vol. 9597 of Lecture Notes in

Computer Science, pp. 525–540, Springer International Publishing, Cham, Switzerland, 2016.

[10] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience in Super Mario Bros," in Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games (CIG), pp. 132–139, Milano, Italy, September 2009.

[11] G. N. Yannakakis and J. Hallam, "Game and player feature selection for entertainment capture," in Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games, pp. 244–251, Honolulu, Hawaii, USA, April 2007

[12] N. Shaker, G. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," in Proceedings of the 6th AAAI Conference on Artifcial Intelligence and Interactive Digital Entertainment, AIIDE 2010, pp. 63–68, Stanford, Calif, USA, October 2010.

[13] Cowley, B., Charles, D. Behavlets: a method for practical player modelling using psychology-based player traits and domain specific features. *User Model User-Adap Inter* **26**, 257–306 (2016). https://doi.org/10.1007/s11257-016-9170-1

[14] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience for content creation," IEEE Transactions on Computational Intelligence and AI in Games, vol. 2, no. 1, pp. 54–67, 2010.

[15] M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin, "Polymorph: A model for dynamic level generation," in Proceedings of the 6th AAAI Conference on Artifcial Intelligence and Interactive Digital Entertainment, AIIDE 2010, pp. 138–143, Stanford, Calif, USA, October 2010.

[16] L. V. Carvalho, A. V. M. Moreira, V. V. Filho, M. T´ulio, C. F. Albuquerque, and G. L. Ramalho, "A Generic Framework for Procedural Generation of Gameplay Sessions," in Proceedings of the SB Games 2013, XII SB Games, S˜ao Paulo, Brazil, 2013.

[17] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Online adaptation of game opponent AI with dynamic scripting," International Journal of Intelligent Games & Simulation, vol. 3, no. 1, pp. 45–53, 2004.

[18] Z. Simpson, "The In-game Economics of Ultima Online," in Proceedings of the Game Developers Conference, San Jose, Calif, USA, 2000.

[19] J. Togelius, R. DeNardi, and S. M. Lucas, "Making racing fun through player modeling and track evolution," in Proceedings of the Workshop Adaptive Approaches Optim. Player Satisfaction Comput. Phys. Games, p. 70, 2006

[20] K. Lohse, N. Shirzad, A. Verster, N. Hodges, and H. F. Van der Loos, "Video Games and Rehabilitation," Journal of Neurologic Physical Terapy, vol. 37, no. 4, pp. 166–175, 2013

[21] Hallifax, S.; Serna, A.; Marty, J.; Lavoué, E. Adaptive Gamification in Education: A Literature Review of Current Trends and Developments. *Lect. Notes Comput. Sci.* **2019**, *11722*, pp. 294–307

[22] Dalponte Ayastuy, M.; Torres, D.; Fernández, A. Adaptive gamification in Collaborative systems, a systematic mapping study. *Comput. Sci. Rev.* **2021**, *39*, 100333

[23] Denden, M.; Tlili, A.; Essalmi, F.; Jemni, M. Does personality affect students' perceived preferences for game elements in gamified learning environments? In Proceedings of the IEEE 18th International Conference on Advanced Learning Technologies, ICALT 2018, Mumbai, India, 9–13 July 2018; pp. 111–115

[24] Borges, S.; Mizoguchi, R.; Durelli, V.H.S.; Bittencourt, I.; Isotani, S. A link between worlds: Towards a conceptual framework for bridging player and learner roles in gamified collaborative learning contexts. In *Advances in Social Computing and Digital Education, Croatia*; Koch, F., Koster, A., Primo, T., Guttmann, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 19–34

[25] Škuta, P.R.; Kostolányová, K. Adaptive approach to the gamification in education. In Proceedings of the European Conference on Technology Enhanced Learning, Transforming Learning with Meaningful Technologies, Delft, The Netherlands, 16–19 September 2018; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; p. 367

[26] Barata, G.; Gama, S.; Jorge, J.; Gonçalves, D. Gamification for smarter learning: Tales from the trenches. *Smart Learn. Environ.* 2015, *2*, 1–23