4th INTERNATIONAL CONFERENCE ON DATA ANALYTICS AND MANAGEMENT (ICDAM-2023) 23rd - 24th June 2023

A comparative analysis of genetic algorithm parameters for Flappy-Bird game automation.

1. Dr. Neelam Sharma
Assistant Professor, Department of Artificial Intelligence and Machine Learning,
Maharaja Agrasen Institute of Technology (MAIT), New Delhi 110082.

Email-neelamsharma@mait.ac.in

2. Kishan Payadi
Student , Department of Artificial Intelligence and Machine learning, MAIT.
Email -payadikishan@gmail.com

3. Amar Singh
Student , Department of Artificial Intelligence and Machine learning, MAIT.
Email- singh.amarraghuvanshi@gmail.com

4. Sahil Verma
Student , Department of Artificial Intelligence and Machine learning, MAIT.
Email- vermasahil1203@gmail.com

Corresponding author(s). Phone Number: +91 8448876965;
Contributing authors: +91 9454098805; +91 8840819053;
These authors contributed equally to this work.

Table of Content

Abstract

Neuroevolutionary algorithms have gained significant attention as a promising approach to solving complex problems. In this research, we present a parameter analysis of neuroevolution for the Flappy Bird Genetic Algorithm using the NEAT (NeuroEvolution of Augmenting Topologies) library. The main objective of this study is to explore the performance analysis of the NEAT algorithm by varying its parameters. To achieve this, the Flappy Bird game was chosen as a testbed. The NEAT algorithm was employed to train an artificial neural network (ANN) to play the game autonomously. By implementing a fitness function that evaluates the game performance, the NEAT algorithm evolves the structure and weights of the ANN to optimize its gameplay strategy. To analyze the performance of the NEAT algorithm, we employed popular data analysis tools such as NumPy, Pandas, and Matplotlib to investigate the effects of parameter variations on the neuroevolutionary process. These parameters include population size, mutation rate, and crossover method. By systematically adjusting these parameters, the impact on the algorithm's convergence, solution quality, and computational efficiency was assessed. The significance of this research lies in the potential applications of automated game-playing and the understanding gained through performance analysis. The findings contribute to the field of evolutionary computation and highlight the adaptability and robustness of the NEAT algorithm in solving complex problems.

Keywords

Flappy-Bird, Genetic Algorithm, Neuroevolution, NEAT, Trade-off between Exploration and Exploitation.

Introduction

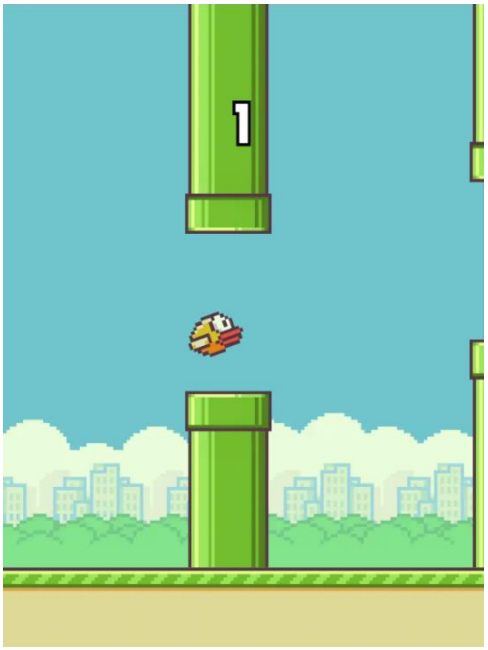

Fig1. – Flappy bird game

Flappy Bird, developed by Dong Nguyen, is a popular mobile game that gained immense popularity due to its minimalist design, addictive gameplay, and competitive nature. Players control a bird avatar and navigate it through gaps in pipes, earning points for successful passages. The game's high difficulty and addictive nature contributed to its success. Flappy Bird went viral in early 2014 and was abruptly removed by the developer, sparking controversy and leading to the emergence of numerous clones. The game generated significant ad revenue and became a social media sensation. Flappy Bird's impact extended beyond gaming, raising discussions on viral games, addictive mobile gaming, and intellectual property rights. Despite its removal, the game's legacy persists, with attempts to revive it and its use as a platform for AI and machine learning research. Flappy Bird remains a significant milestone in the history of mobile gaming.

Genetic algorithm

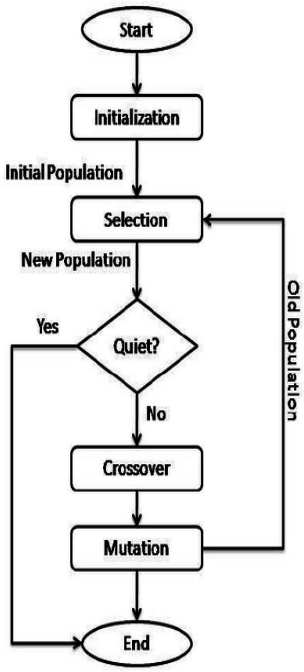

Fig2: Flowchart of Genetic Algorithm [11]

A genetic algorithm (GA) is an optimization algorithm inspired by natural selection and evolution. It is used to solve complex optimization problems, especially when traditional algorithms struggle. GAs are effective in large, non-linear, or poorly understood solution spaces. Introduced by John Holland in the 1960s and 1970s, genetic algorithms mimic natural evolution and genetics in a computational context. Early research focused on theoretical foundations and mathematical models, exploring selection, crossover, and mutation as genetic operators. In the 1980s, David Goldberg and John Koza made practical contributions to GAs. Goldberg introduced binary strings as chromosome representations and investigated selection mechanisms. Koza pioneered genetic programming, applying GAs to evolve computer programs.

The key components of a genetic algorithm include the representation of individuals as potential solutions, initialization of a random population, evaluation using a fitness function, selection based on fitness, reproduction through crossover and mutation, replacement of the previous generation, and termination based on specific criteria.

Genetic algorithms draw from Charles Darwin's theory of natural selection, using chromosomes and genes to represent solutions. A fitness function evaluates individual solutions, guiding selection. Crossover combines genetic material from parents, mimicking recombination, while mutation introduces random changes. Reproduction and replacement generate new populations, evolving solutions over generations.
Overall, genetic algorithms provide a powerful optimization approach by emulating the principles of evolution and genetics in problem-solving. [1] [2].

Neuroevolution

Neuroevolution combines neural networks and evolutionary algorithms to train and optimize artificial neural networks. Instead of manually designing neural network structures, it allows networks to evolve through a process like natural selection. A population of neural networks is created with random or predefined configurations. These networks are evaluated on a task using a fitness function. The fittest networks are selected, and genetic operators like mutation and crossover are applied to create new offspring. This process is repeated over multiple generations to improve network performance. Neuroevolution explores a large search space and can solve complex problems that traditional methods may struggle with. It automates network design, handles complex problems, adapts to dynamic environments, enables transfer learning, and utilizes parallelization for efficient computation.
In summary, neuroevolution leverages the power of evolution to optimize neural network architectures and parameters, making it valuable in various domains and applications. [3] [4]

Neat algorithm

NEAT (NeuroEvolution of Augmenting Topologies) is a neuroevolution algorithm designed for evolving artificial neural networks (ANNs) with dynamic and complex structures. It combines principles from neural networks and evolutionary algorithms to optimize both network architectures and connection weights.

➢ Key Components of NEAT:

- Genome Representation: ANNs are represented as genomes consisting of nodes (neurons) and connections (synapses) with innovation numbers to track historical information.
- Historical Marker and Innovation Numbering: NEAT uses markers and numbers to preserve innovation history during evolution, aiding crossover and tracking successful connections.
- Speciation: NEAT employs speciation to maintain diversity by grouping similar genomes into species. This allows independent evolution within species and prevents premature convergence.
- Fitness Evaluation and Selection: The fitness function assesses network performance, and selection methods determine individuals for reproduction based on fitness.
- Crossover and Mutation: Genetic operators, such as crossover and mutation, create offspring by inheriting genes from parent genomes and introducing small random changes.
- Complexification and Simplification: NEAT enables the growth and simplification of networks by adding or removing nodes and connections through mutation.

➢ Advantages and Applications:

- Incremental Growth of Complexity: NEAT facilitates the gradual development of increasingly complex neural networks from minimal initial structures.
- Topology and Weight Evolution: NEAT simultaneously evolves network architectures and connection weights to discover optimal structures for specific tasks.
- Promotes Innovation: Historical markers and innovation numbering encourage the emergence and spread of novel solutions within the population.
- Adaptability to Dynamic Environments: NEAT allows structural changes to adapt neural networks in real time to dynamic or changing environments.
- Real-world Problem Solving: NEAT has been successfully applied to various domains, including control systems, robotics, game playing, and optimization tasks. [5] [6]

➢ Additional Parameters of NEAT:

- Population Size: Refers to the number of individuals (genomes) in the population. A larger population promotes diversity but requires more computational resources, while a smaller population may converge faster but risk reduced diversity.
- Survival Threshold: Determines the percentage of top-performing species allowed to reproduce and survive. It impacts diversity preservation and convergence rate by controlling the selective pressure applied to the population.
- Fitness Criterion: Measures the performance or quality of each genome using a problem-specific evaluation. For example, in a game-playing scenario, it could be the achieved score. Genomes are selected for reproduction based on their fitness.
- Elitism: Preserves the best-performing individuals from one generation to the next without any modification. It helps maintain high-quality solutions and speeds up convergence.
- Activation Default/Mutation/Rate: Activation Default refers to the initial activation function applied to nodes. Activation Mutation changes the activation function of a node through variation. The activation Mutation Rate determines the probability of activation function mutation during evolution. They explore different activation functions and network configurations.
- FeedForward: A boolean value determining whether the neural network should have a feed-forward architecture. Useful for tasks where feedback connections are not required.
- Compatibility_disjoint_coefficient/compatibility_weight_co-efficient: Coefficients used in calculating compatibility distance between genomes. They measure differences in disjoint genes and weights, respectively, and help determine crossover compatibility and speciation.
- Aggregation Default: Defines the default method for combining outputs of neurons within a network. It affects how information from different neurons is processed and combined.

- Stagnation: The maximum number of generations a species can remain stagnant without fitness improvement. Stagnant species may face extinction, promoting diversity and preventing stagnation in the evolutionary process.
- These parameters control aspects such as population diversity, selective pressure, activation function variations, network architecture, crossover compatibility, and preservation of superior individuals.
- Overall, NEAT is a powerful algorithm that addresses the challenge of evolving neural networks with dynamic topologies. Its ability to adapt and optimize network structures has contributed to its widespread use in various practical applications.

Literature survey

In 2021, Katoch, Chauhan, and Kumar conducted a comprehensive review of genetic algorithms (GAs) titled "A review of the genetic algorithm: past, present, and Future." Their study provided insights into the historical development, current state, and prospects of GAs. They explored various applications and advancements of GAs in diverse fields. The review highlighted the effectiveness of GAs in solving complex optimization problems and their potential for further enhancements. Katoch et al.'s work served as a valuable resource for understanding past achievements, current trends, and future directions of genetic algorithms. [1]

In their 2010 paper titled "Genetic Algorithm: Review and Application," Kumar, Husain, Upreti, and Gupta conducted a comprehensive review of genetic algorithms (GAs) and their applications. The survey covered key topics such as the principles, techniques, and advancements in GAs. It explored various parameters including selection, crossover, and mutation operators, while highlighting the importance of fitness evaluation and population management. The authors also discussed successful applications of GAs in fields such as engineering, economics, and computer science. This literature survey serves as a valuable resource for understanding and applying GAs in solving complex optimization problems. [2].

In 2002, Stanley and Miikkulainen introduced NEAT, a method that combines genetic algorithms and neural networks. NEAT addresses the challenge of evolving complex neural networks by gradually adding nodes and connections over generations. It allows for the evolution of both network structure and connection weights, surpassing the limitations of fixed-structure networks. Experimental results demonstrated NEAT's superior performance compared to traditional methods, leading to its adoption in game playing, robotics, and control systems. NEAT's impact inspired further research and extensions, exploring multiobjective optimization, hyper neat, and recurrent networks. Stanley and Miikkulainen's work revolutionized neuroevolution, offering a new approach and driving advancements in the field. [3].

The research paper by Deepkumar, Nithin, and Vani Vasudevan titled "Neuro-Evolution in Flappy Bird Game" presents a concise literature review on the use of neuro-evolution in game playing. The authors outline the design of the neural network, the representation of the game state, and the fitness function used to evaluate the agent's performance. They employ a genetic algorithm to evolve the neural network's weights and biases over multiple generations. Experimental results demonstrate that the neuro-evolutionary agent outperforms both random and handcrafted agents in terms of the average score achieved in Flappy Bird. The paper also discusses limitations and potential future directions for improving the agent's performance. [4].

The research paper titled "Innovative Application of Genetic Algorithms in Computer Games" by Tao, Jun, Gui Wu, Zhentong Yi, and Peng Zeng provides a concise literature survey on the utilization of genetic algorithms (GAs) in computer games. The paper explores applications such as game level generation, character behavior design, strategy optimization, and game balancing using GAs. The research highlights the advantages and successful implementations of GAs in computer games, contributing valuable insights for future developments. [5].

In their 2019 study on the performance analysis of a Flappy Bird playing agent, Mishra, Kumawat, and Selvakumar considered several important parameters. These parameters included the neural network architecture, genetic algorithm parameters, fitness function, exploration vs. exploitation trade-off, training data and environment, and evaluation metrics. By examining and optimizing these parameters, the researchers aimed to enhance the agent's gameplay strategy and evaluate its performance. Their study provided valuable insights into the role of these parameters in analyzing the Flappy Bird playing agent's performance. [7].

In their 2013 study, Mohabeer and Soyjaudah focused on improving the performance of NEAT (NeuroEvolution of Augmenting Topologies)-related algorithms by reducing complexity in the search space. They employed several techniques for complexity reduction, including fitness function refinement, pruning irrelevant connections, and speciation enhancement.By refining the fitness function, the researchers aimed to provide more accurate guidance for the evolutionary process, eliminating unnecessary search operations. They also explored methods to identify and remove irrelevant connections in neural network topologies, reducing the complexity of the search space. Additionally, enhancing the speciation process ensured a better distribution of individuals, avoiding redundant search operations within species.Through these techniques, Mohabeer and Soyjaudah sought to streamline the search process and improve the efficiency of NEAT-related algorithms. Their study contributed valuable insights into complexity reduction strategies, offering potential enhancements to the performance of NEAT and related algorithms. [8].

Methodology

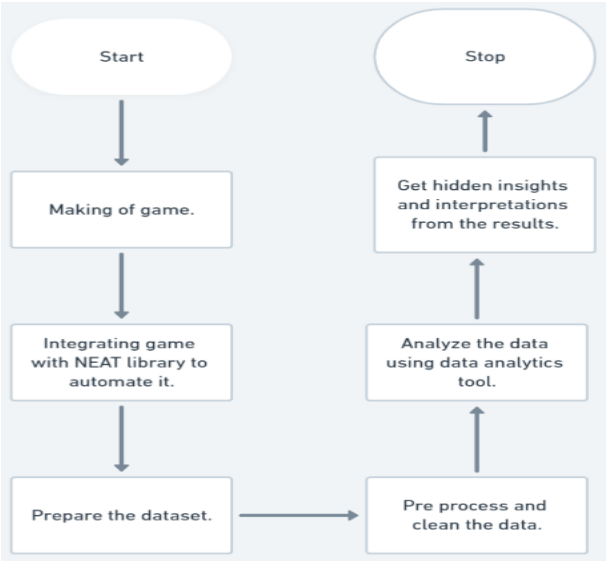

Fig3: flowchart of workflow

In this project, the main objective was to design and implement the Flappy Bird game environment and automate the gameplay using

a genetic algorithm. The development process started by utilizing Python's pygame library to develop the Flappy Bird game itself, creating a fully functional game environment complete with the bird agent, pipes, and scoring mechanism. To automate the gameplay, the NEAT (NeuroEvolution of Augmenting Topologies) library was employed, which allowed for the implementation of a genetic algorithm. The bird agents were represented as neural networks with two layers, an input layer, and an output layer. The input layer consisted of neurons that encoded information about the bird's position, distance from the top pipe, and distance from the bottom pipe. The output layer consisted of a single neuron that determined the bird's actions based on various weights, biases, and activation functions. Throughout the development process, an iterative software development method was employed, enabling incremental improvements and refinements to be made to the game and its automation. To evaluate the performance of the game, fitness functions or evaluation metrics were defined to measure the effectiveness of the bird agents. Additionally, different activation functions such as tanh, sin, sigmoid, and soft plus were tested to analyse their impact on the gameplay. Parameters such as fitness criterion, mutation rate, and population size were carefully considered to assess the performance of the genetic algorithm. Visual representations in the form of graphs were used to illustrate the relationship between these parameters and the time taken to achieve a score of 100. Experiments were conducted by systematically varying one parameter at a time while keeping others constant. The experiments were terminated when a bird agent reached a score of 100. The number of generations or specific termination conditions was used to control the experiments, and the time required to reach the target score was recorded for each parameter configuration. For the analysis and presentation of data, powerful tools like NumPy and Pandas were employed for data pre-processing and manipulation. Statistical methods were applied to analyse the experimental results, allowing for the calculation of average fitness, convergence speed, and diversity of solutions. To present the findings, visualization tools such as Seaborn and Matplotlib were utilized to generate informative graphs. Outliers in the data were identified using techniques like boxplots, and appropriate measures were taken to handle them, including manual removal if necessary. It is essential to acknowledge the limitations and challenges encountered during the experiments. The process of manually recording the dataset by observing and adjusting parameter values was found to be time-consuming. Furthermore, the absence of a hidden layer in the neural network design, although contributing to easier training, may have increased the risk of overfitting the model.

Results and interpretations

1. Population size vs time required to reach a score of 100

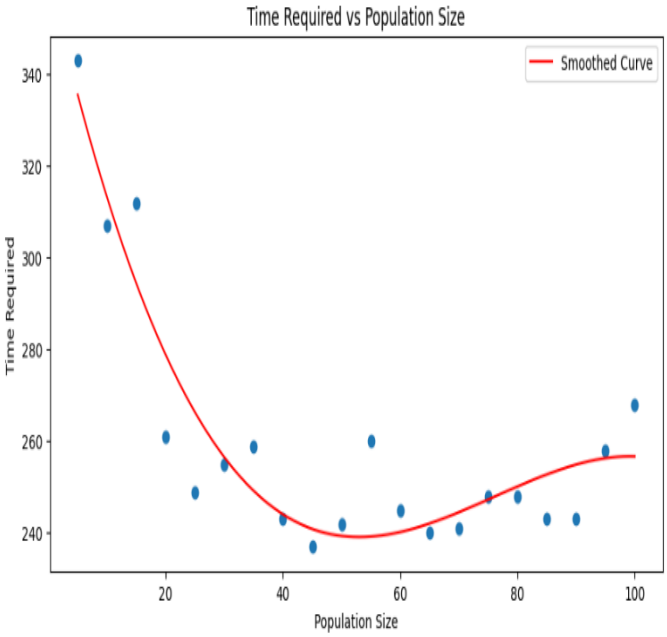

Fig4: Graph b/w Time Required vs Population Size

Interpretation:

- Efficiency Optimization: Optimal population size minimizes the time required, striking a balance between too few or too many individuals.
- Exploration and Exploitation Trade-off: There is an efficient state where increasing population size no longer significantly improves the time required. Balancing exploration and exploitation is crucial, considering computational overhead.
- Resource Allocation: Allocating resources to a population size near the local minimum ensures efficient utilization, improving performance in terms of time required.
- Algorithm Performance Evaluation: The curve shape provides insights into the algorithm's sensitivity to population size, allowing comparisons and guiding improvements for similar problems.

2. Fitness criterion vs Time required to reach a score of 100

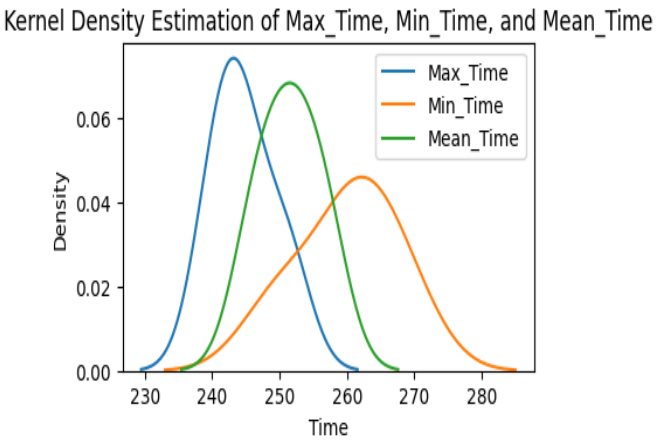

Fig5:- Graph b/w Density vs Time

Interpretations:

- Vertical Stretch: The vertical stretching of the density plot for the min fitness criterion suggests less variation or spread in the time taken to reach a score of 100. This implies a more consistent and predictable performance with minimizing the fitness criterion, although there may be outliers.
- Time of Peak: The order of the peaks (max < mean < min) indicates that the algorithm reaches its peak performance earliest when using the max fitness criterion, followed by the mean, and finally the min fitness criterion. This suggests that the max fitness criterion allows for the quicker achievement of the target score on average.

3. Activation default

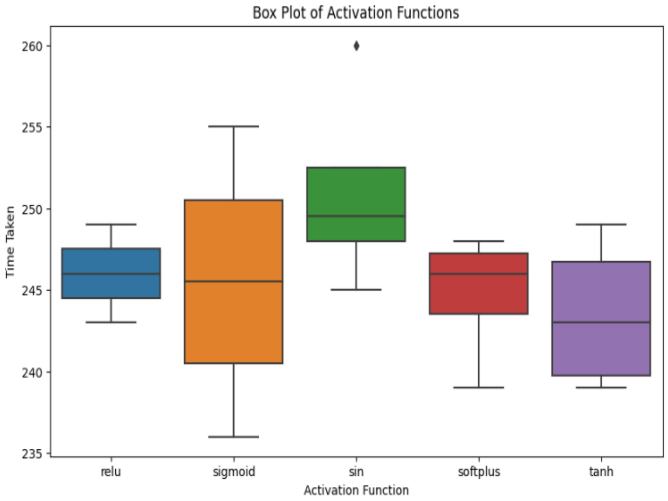

Fig6: Graph b/w Time Taken vs Activation function

Interpretation of the result

- ❖ Order of Means: The mean time required to reach a score of 100 for each activation function follows the order: tanh < sigmoid < soft plus < relu < sin. This suggests that, on average, the tanh activation function requires the least amount of time, followed by sigmoid, soft plus, relu, and sin.
- ❖ Symmetry and Compression:
  - Relu: The relu activation function shows a symmetrical and compressed box plot, indicating balanced and tightly distributed data with consistent time requirements.
  - Sigmoid: The sigmoid activation function exhibits a symmetrical box plot but without compression, suggesting some variability in the time required.
  - Tanh: The tanh activation function displays compression within the box, indicating a clustering of data points around the median and relatively consistent time requirements.
  - Softplus and Sin: The box plots for soft plus and sin activation functions are not symmetrical, suggesting an uneven distribution of the time required, potentially indicating skewness or outliers in the data.

These interpretations highlight the impact of the activation function choice on the time required to reach a score of 100. The tanh activation function tends to require the least time on average, while relu shows a consistent and compressed distribution. Sigmoid, soft plus, and sin exhibit variability and non-symmetrical distributions, indicating potential differences in their performance and behavior.

4. Activation mutation rate vs time required to reach a score of 100

Default activation = tanh
Activation options = {sigmoid, soft plus, tanh}

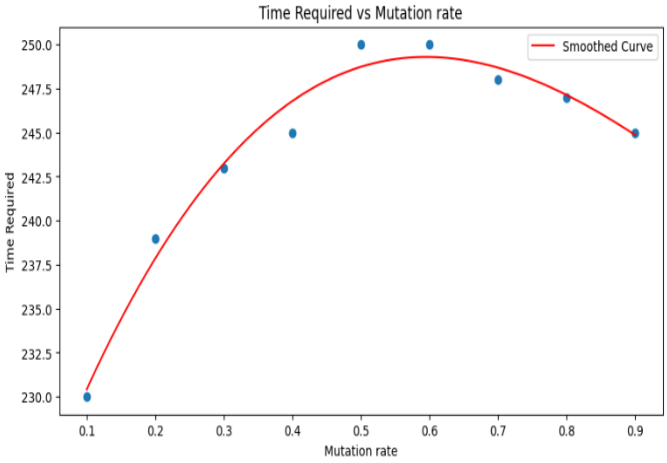

Fig7: Graph b/w Time Required vs Mutation rate

Interpretations:

- Exploration vs. Exploitation: In the NEAT algorithm, the mutation rate determines the balance between exploration and exploitation. Lower mutation rates prioritize exploitation, refining current solutions for faster convergence and decreased time required to reach the target score.
- Overfitting: Increasing the mutation rate beyond a certain point promotes exploration, introducing more variability and randomness. This can lead to longer convergence times, representing a trade-off between exploration and exploitation. Excessive exploration may result in suboptimal solutions and prolonged convergence.
- Optimal Mutation Rate: The decrease in time required after a mutation rate of 0.5 suggests the existence of an optimal range for mutation rate. This range effectively balances exploration and exploitation, allowing the algorithm to explore diverse solutions while exploiting promising ones. Identifying and utilizing this optimal mutation rate can lead to faster convergence and improved performance in terms of the time required to reach the target score.

5. Elitism vs time required to reach a score of 100

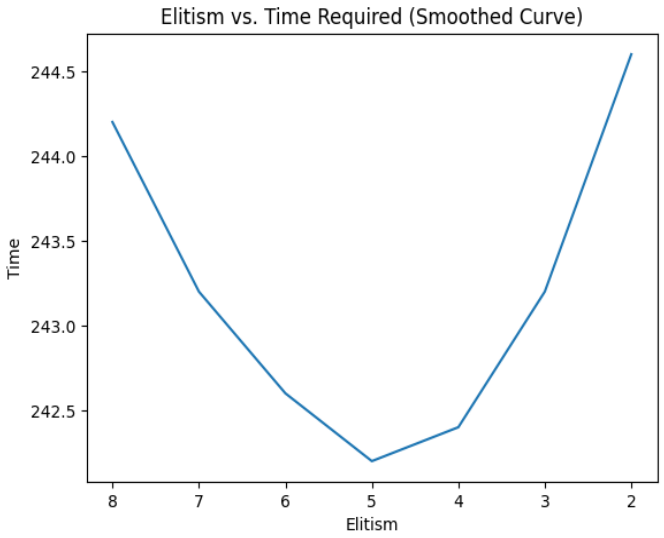

Fig8: Graph b/w Elitism vs Time Required

Interpretations:

- Balancing Exploration and Exploitation: Elitism in the NEAT algorithm preserves the best-performing individuals without modifications, promoting the exploitation of high-quality solutions. Increasing elitism leads to more exploitation and potentially faster convergence. However, extreme values can limit diversity and hinder exploration, prolonging convergence times.
- Optimal Balance for Performance: The local minimum at an elitism value of 5 indicates an optimal balance between exploration and exploitation. This configuration maintains diversity while benefiting from elitism's advantages. It efficiently explores the solution space and exploits promising individuals, resulting in reduced convergence times.
- Sensitivity to Elitism: The curve's shape shows the algorithm's sensitivity to elitism changes. Deviating from the optimal value of 5 (lower or higher) can increase convergence times. Lower values reduce exploitation, while higher values limit diversity and exploration. Selecting the appropriate elitism value is crucial for efficient optimization and minimizing the time required to reach the target score.

I. Survival threshold vs time to reach a score of 100.

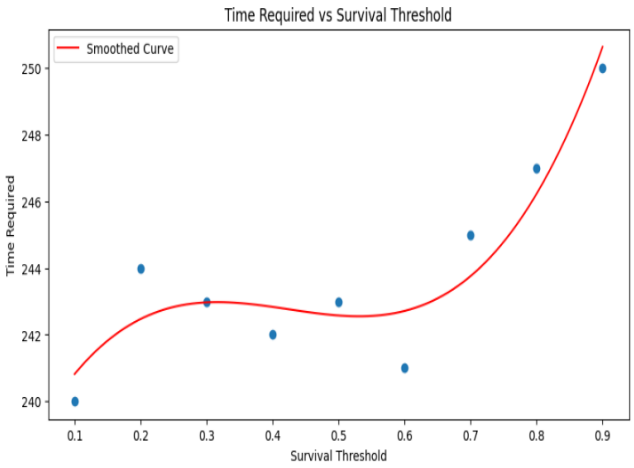

Fig9: Survival threshold vs time to reach a score of 100

Interpretations:

- Survival Selection Pressure: The survival threshold determines the level of selection pressure in NEAT. A lower threshold leads to the stringent selection, eliminating weaker individuals quickly and resulting in faster convergence to the target score.
- 2. Balancing Exploration and Exploitation: Stabilizing time between 0.3 and 0.6 survival thresholds indicates a balance between exploration and exploitation. This range allows for moderate selection pressure, enabling sufficient exploration while retaining promising individuals and leading to stable optimization with consistent convergence times.
- Suboptimal Solutions: Beyond the 0.6 survival threshold, the exponential increase in time required suggests difficulty in finding suitable solutions. A higher threshold relaxes selection pressure, allowing weaker individuals to persist and explore a larger space of suboptimal solutions before eventually converging to better solutions, resulting in longer convergence times.

Conclusion

Our research has demonstrated the significance of genetic algorithm parameters in the Flappy-Bird game automation using the NEAT library. Through our analysis, we have gained valuable insights into the impact of parameter settings on the efficiency and effectiveness of the algorithm. The results indicate that selecting appropriate genetic algorithm parameters is crucial for achieving optimal performance. We have observed that certain configurations, such as a moderate mutation rate and a balanced approach to exploration and exploitation, lead to faster convergence and improved performance. These findings have practical implications for game developers and researchers working on similar applications. By optimizing the algorithm based on our recommendations, they can save computational resources and development time while achieving desirable outcomes. Furthermore, the insights gained from our research can be generalized to other game-playing scenarios and non-gaming domains that employ genetic algorithms and neuroevolutionary techniques. In summary, our research contributes to the understanding of how genetic algorithm parameters impact game automation. It provides valuable insights and recommendations for parameter selection, guiding practitioners and researchers in optimizing the training process. Future research can build upon our work by exploring additional variables and techniques to further enhance the efficiency and effectiveness of genetic algorithms in game automation.

Table 1:

Parameter name	Optimal value	Justification of Optimal value
Mutation rate	0<rate<0.1	Achieves balance between exploration and exploitation.
Elitism	5	Maintains diversity
Survival Threshold	0.5	Moderate selection pressure and balance between exploration and exploitation.
Fitness Criterion	Max	Focusing of choosing the best performing birds.
Population size	50	Adequate population size to maintain diversity and facilitate exploration.
Activation default	Tanh	Tanh activation function has shown to perform well in similar scenarios.

Future scope.

I. Contribution to the Field: Our research paper contributes to the field by analysing the effects of changing variables in the NEAT algorithm on game performance. This provides valuable insights into how different parameter settings impact the algorithm's efficiency and effectiveness. The knowledge gained can guide future researchers and practitioners in optimizing the algorithm for similar game-playing scenarios or other domains, improving overall performance and outcomes. [3]

II. Insights into Optimization: By studying the impact of various variables, our research paper offers insights into the optimization process of the NEAT algorithm. We specifically examine performance metrics, such as the time required, to identify optimal settings and trade-offs. Understanding these factors enables researchers and practitioners to fine-tune the algorithm, achieving desired outcomes more effectively and efficiently. [9]

III. Practical Implications: Our research paper has practical implications for game developers and researchers in related fields. It provides guidelines on parameter selection, aiding in the optimization of the training process and facilitating the achievement of a score of 100 more efficiently. Implementing these findings can significantly save computational resources and development time, enhancing the practicality and feasibility of using the NEAT algorithm in real-world applications. [5]

IV. Generalizability: Although our research focuses on a specific game and performance metrics, the insights gained from analysing NEAT variables have broader implications. The findings can be generalized to other game-playing scenarios or non-gaming domains that utilize neuroevolutionary techniques. This expands the applicability of our research, extending its relevance beyond the specific context of the game studied.

V. Future Research Directions: Our research paper opens new avenues for future research in the fields of neuroevolution and game AI. Researchers can build upon our work by exploring additional variables, investigating different fitness criteria, considering alternative activation functions, or incorporating other optimization techniques. These directions can lead to further advancements, improving the effectiveness and efficiency of training algorithms in these domains. [6].

References

[1] Stanley, Kenneth O., and Risto Miikkulainen. "Evolving neural networks through augmenting topologies." Evolutionary computation 10, no. 2 (2002): 99-127.

[2] Mohabeer, Heman, and K. M. S. Soyjaudah. "Improving the Performance of NEAT Related Algorithm via Complexity Reduction in Search Space." In Distributed Computing and Artificial Intelligence: 10th International Conference, pp. 1-7. Springer International Publishing, 2013.

[3] Katoch, Sourabh, Sumit Singh Chauhan, and Vijay Kumar. "A review on genetic algorithm: past, present, and future." Multimedia Tools and Applications 80 (2021): 8091-8126.

[4] Mishra, Yash, Vijay Kumawat, and K. Selvakumar. "Performance analysis of flappy bird playing agent using neural network and genetic algorithm." In Information, Communication and Computing Technology: 4th International Conference, ICICCT 2019, New Delhi, India, May 11, 2019, Revised Selected Papers 4, pp. 253-265. Springer Singapore, 2019.

[5] Kumar, Manoj, Dr Mohammad Husain, Naveen Upreti, and Deepti Gupta. "Genetic algorithm: Review and application." Available at SSRN 3529843 (2010).

[6] Deepkumar, Nithin, and Vani Vasudevan. "Neuro-Evolution in Flappy Bird Game." In 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), pp. 1-5. IEEE, 2022.

[7] Tao, Jun, Gui Wu, Zhentong Yi, and Peng Zeng. "Innovative Application of Genetic Algorithms in the Computer Games." In *2021 33rd Chinese Control and Decision Conference (CCDC)*, pp. 2197-2200. IEEE, 2021.

[8] Immanuel, Savio D., and Udit Kr Chakraborty. "Genetic algorithm: an approach on optimization." In *2019 international conference on communication and electronics systems (ICCES)*, pp. 701-708. IEEE, 2019.

[9] Kumar, Shailender, Gaurav Khatri, Gurvinder Singh, and Hardik Gupta. "Artificial Intelligent Player Character Using Neuroevolution of Augmenting Topologies and Neural Networks." In *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, pp. 1-6. IEEE, 2022.

[10] Naud Ghebre, Ethan Jen, Matthew McBrien, Arihan Shah, and Nahom Solomon. "Flappy Bird: Neuroevolution vs NEAT." *mmcbrien. com* (2017).

[11] M. Abdullah, S. R. A. Rahim, and A. S. Jaafar, "Used Genetic Algorithm for Support Artificial Neural Network in Intrusion Detection System," in 2013 IEEE Conference on Information Technology and Applications (ICITA), Sydney, Australia, 2013, pp. 342-347. doi: 10.1109/ICITA.2013.6750217