Lineup Mining and Balance Analysis of Auto Battler

Jiayu Xu*[1,2], Shifeng Chen[1], Like Zhang[2], Junle Wang[2]
[1]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, 518000, China
[2]Tencent, Shenzhen, 518000, China
*Corresponding author
e-mail: jiayuxu@tencent.com

2020 IEEE 6th International Conference on Computer and Communications (ICCC) | 978-1-7281-8635-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICCC51575.2020.9345074

***Abstract*—Auto battler is currently one of the most popular types of video games on the market. In such type of games, players set up their own lineup of characters which combat automatically with other competitors. There are numerous kinds of chess pieces, lineups and bonuses in the game, which make the balance problem of game design extremely critical and challenging. In this paper, we use Chess Rush to investigate the methodology of evaluating the strength standards of lineups in auto battler games, and build a system for implementing such measurement. A neural network model is firstly trained for quick strength evaluation of the lineup. The genetic algorithm is then adopted and modified for the purpose of lineup mining of auto battler. Finally, strong lineups can be obtained under certain constraints. In this way, game developers can complete a balanced analysis before the game goes online. Moreover, the proposed approach can be generalized to balance analysis tasks of other games.**

Keywords-auto battler game, lineup evaluation model, genetic algorithm, balanced analysis, chess rush

I. INTRODUCTION

Auto battler genre video game is based on the DotA 2 custom map constructed in early 2019. In this game, eight players can purchase chess pieces from the public pool in each round and place them on the board. Through a reasonable arrangement of lineups, a player can defeat the other players' lineup. The battle process is automatic. Each piece in the game is sorted into different classes and races, including warriors, mages, goblins, trolls, undead, etc. If the chess pieces on the board with the same races or classes reach a certain number, a bonus will be triggered, such as the warrior receiving a buff for increase in the armor for all allied warriors. Three identical one-star pieces can be combined into a single two-star piece, and three identical two-star pieces can be combined into a single three-star piece. The multiple stars of the chess pieces can significantly increase the combat effectiveness in the game. Games presently in the market include Tencent's King Glory Auto Battler and Chess Rush, DRodo Studio's DotA Auto Battler, and Valve's DotA Underlords. Consequently, lineup mining and balance analysis of auto battlers have become research problems for game developers.

In auto battler games, balance is among the most important factors affecting the player experience. A well-balanced auto battler game has a variety of strong lineups. Players can choose the right lineup based on the draw. If the players cannot draw specific pieces, they cannot form a more powerful lineup, which affects their gaming experience. The diversified and balanced game lineup gives players a sense of novelty. It provides experienced players with more operating space, which can maintain good game activity. If there is a clearly unbalanced lineup in the auto battler game, most players will choose the same lineup, hence it is difficult for all players to make up the unbalanced lineup. This results in significant frustration for players. Therefore, to maintain the player's retention rate, it is necessary to adopt an appropriate method to study the balance in the auto battler game.

The balance in the auto battler game is extensively important, however there are more difficulties to overcome. First, the auto battler generally contains almost 200 chess pieces of different levels, and more than 20 classes and races. Once a chess piece or bonus is strengthened, it affects numerous lineups, and the specific impact is difficult to quantify and analyze. Second, the auto battler is a mode in which eight players battle each other through multiple rounds of games, hence it is necessary to comprehensively measure the strength of the lineup. Third, different chess pieces have different skills and level bonus effects, which are difficult to quantify. Finally, self-checking involves a large number of parameters, and the traditional method takes a long time to traverse, which makes it is difficult to find the optimal lineup. Therefore, a better method for lineup mining and balance adjustment is required.

The goal of this study is to find a strong lineup under different constraints through lineup mining and to adjust the related chess pieces and bonuses by analyzing the abnormal lineup. Further, the lineup mining algorithm can be used to make real-time recommendations. All the data and experiments in this paper take Tencent's Chess Rush game as an example.

Chess Rush is a turn-based auto battler game of eight players by Tencent. Each player competes against the other seven players to achieve the final victory. In general, there are dozens of different kinds of chess pieces in the auto battler. Furthermore, there are one-star, two-star, and three-star pieces for each chess piece. The two-star piece is automatically synthesized by three identical one-star pieces, whereas the three-star piece is synthesized by three identical two-star pieces. Each piece contains two different bonuses of race and class. Once more than a certain number of races or classes are on board at the same time, the bonus will play an important role in these pieces.

This game has achieved success in the international market, with numerous players in Korea, Thailand, and other countries. The purpose of this research is to find a more

978-1-7281-8635-1/20/$31.00 ©2020 IEEE

powerful lineup combination under multiple constraints by employing lineup mining. Chess pieces and bonus compositions in the strong lineup are analyzed, to refer to the balance adjustment.

II. BACKGROUND

About a year has passed since the emergence of auto battler games, and as game developers and researchers are still in the exploration stage, there are few studies on the balance analysis. First, the study on the game balance needs to establish a measure of the strength of a single lineup. The most important indicator of lineup strength is the lineup win rate. If the lineup win rate is higher, the strength of the lineup is stronger. Second, according to the lineup strength model, the strongest lineup is automatic. The game developer finally adjusts the balance of the game based on the distribution of the strongest lineup pieces and bonuses.

MOBA genres are among the most popular game genres in the world, and there is significant research on the win rate model of the game [1,2]. Yang et al. used hero features, player features, and combined hero-player features to predict the win rate of DotA2 [3]. Yang's heroes' features contain hero selection, which are heroes' one-hot encoding, attributes, and the win rate. However, it is difficult to describe the relationship between heroes in this hero feature. Thus, the synergy and countering relationship of heroes is difficult to express using these features.

To determine the relationship between heroes, Kalyanaraman et al. used network communities in win rate predictive modeling [4]. Network communities make up part of network science, which models related phenomena. Kalyanaranman used DotA2 matches to build co-occurrence graphs, in a study that identifies the set of heroes that contribute most to victory. At last, they predict the win rate combine regression and genetic algorithm. The result shows that hero composition plays a tremendously important role in the outcome and should not be neglected in matches.

Kinkade, Hodge, Wang et al. have extensively studied the pre-match win rate prediction in MOBA games, and they have achieved relatively good results [5-9]. Although previous algorithms used a variety of methods to consider the heroes' relationships, the traditional model is difficult in terms of considering the chess pieces' relationships in the auto battler. Because the auto battler is a game between multiple players, it is difficult to measure the strength of the lineup in the 1v1 match. Therefore, in this study, the lineup measurement and prediction model is studied for the auto battler.

In addition to researching the lineup strength prediction model, it is also necessary to use lineup strength evaluation to perform lineup mining, and subsequently perform a balanced analysis on the auto battler. At present, studies are being conducted on lineup mining and recommendations for MOBA games, chess, and card games, MMO games, etc. However, line mining for the auto battler remains relatively unexplored. The Monte Carlo tree search (MCTS) is a general term for a class of tree search algorithms that can effectively solve some problems with a large exploration space. MCTS is not a simulation algorithm, as it represents the process of gradually building an asymmetric search tree by randomly deducing the game[10].

The MCTS can be roughly divided into four steps, namely selection, expansion, simulation, and backpropagation [10-13]. Jiang et al. used an FPMCTS algorithm on the Doudizhu card game [14]. They successfully integrate FPMCTS and an inference algorithm into an AlphaZero-like framework. In the FPMCTS search tree for a specific player, there are two types of nodes: decision nodes are the ones where the current player acts, and the remainder are chance nodes. They merge the two opponents' consecutive actions into one chance node.

Zhang et al. proposes an artificial intelligence (AI) fighting strategy generation approach implemented in the turn-based fighting game StoneAge 2 (SA2) [15]. The aim of the study is to develop such AI for searching the logical skills and targets of the player. The approach trained the logistical regression (LR) model and deep neural networks (DNN) individually and combined both outputs at the inference process.

LR, DNN, and MCTS can all be used to solve problems by searching for a large solution space. However, since the game of the auto battler is more constrained, the search space needs to be adjusted under different constraints. Therefore, the above-mentioned search method that requires being re-trained is not suitable for games of the auto battler genre. Therefore, this study investigates the mining algorithm of the auto battler.

III. METHODOLOGY

To solve the problem of balance adjustment, we first need to determine the lineup measurement system. Only a suitable lineup measurement system can provide a basis for lineup mining. Second, we need to find a suitable lineup mining algorithm. This algorithm can mine the strongest lineup under different constraints, such as star rating, cost, number, etc. Finally, statistics are calculated based on the strong lineup that has been mined, and finally, the bonuses and pieces that need to be adjusted are obtained. The overall process is illustrated in Fig. 1.

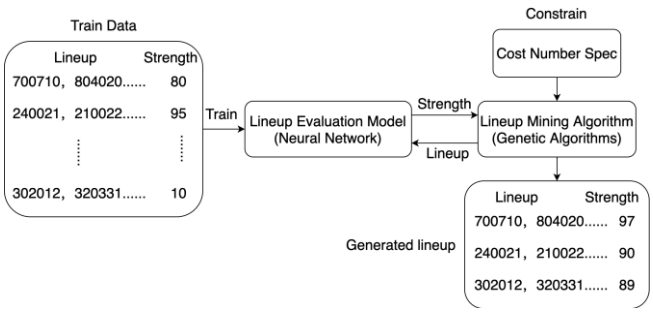

Figure 1. Lineup measurement system

The flow chart indicates that a lineup measurement system needs to be established. This system is used to measure historical or random lineups to obtain the ratings of different lineups. However, in general, these methods require significant calculation, and it is difficult to obtain the results

in real-time. Therefore, a lineup evaluation model is built and trained by inputting lineup strength data. After training, the model achieves quick evaluation of the lineup. A quick evaluation model is key to speed up the lineup mining algorithms.

After obtaining the trained evaluation model, the following step is to perform lineup mining. First, constraints of the mined lineup need to be determined, such as cost, number of chess pieces, and chess stars. After the constraints are input into the lineup mining algorithm, the strongest lineup under the constraint is obtained by using the algorithm. By traversing the strongest lineup under multiple costs, numbers, and constraints, game developers can use the lineups' results to make balance adjustments. For example, a bonus appears more frequently in a strong lineup, and the win rate is higher than other bonuses. Subsequently, the bonus can be adjusted for balance. Lineup combinations can be obtained through an algorithm before a new version of the game goes online, hence avoiding the disadvantage of artificially designed lineups. In addition to the application of balance adjustment, the lineup evaluation and mining system can also be applied to lineup recommendation, player hosting, etc., and it has good application prospects.

A. Lineup Evaluation Model

Because the auto battler has numerous factors, the lineup evaluation model is highly complex. It is difficult to directly apply the evaluation model of MOBA, cards, and chess games. Therefore, the strength of a single lineup needs to be considered comprehensively. A lineup cannot obtain a high rating for restraining a certain lineup. It needs a more balanced performance facing all other lineups, such that it can obtain a higher score. This research obtains players' historical lineups from the database. Lineups with few rounds are not used due too premature elimination of the player in the game to reference. To render lineup sets more representative, the remaining multi-round lineup is sorted and the most frequently used top 1000 lineup sets are employed. The higher the win rate, the stronger the strength of the lineup. When a lineup needs to be evaluated, it is matched with reference lineup sets that contain the top 1000 lineups to calculate the win rate of this lineup. Because it takes almost five minutes to complete the simulation in a computer with a thousand battlers, a distributed cluster is adopted in this calculation. The task is calculating in parallel as shown in Fig. 2, which speeds up the entire process. After the calculation and summary, the win rate obtained is used as the final lineup rating. When the calculated rating of the lineup is close to zero, it implies a lower strength of the lineup. When the calculated rating is close to one, a higher strength of the lineup is implied, which means that it is easier for the player to win in the game. Randomly generated lineups and extract lineups from historical data are calculated for lineup strength. The calculation results are saved to the database for the subsequent model training.

After clarifying the measurement criteria for the auto battler lineup, the following step is to select the features for input of the lineup evaluation model. Feature selection is a key factor affecting the model training results. If inappropriate features are selected, the model will be difficult to fit, and the generalization performance will be poor. Chess pieces are the most critical factor in the game, hence the pieces would be encoded into one-hot encoding, which is input into the model for prediction. Hero attributes such as blood, magic, and attack power are part of the characteristics of the chess pieces. Because of the chess piece skills and bonus, the attributes are dynamically changed. Hence, those attributes are not considered features. In some studies, the synergy and countering of heroes are considered in MOBA games. The win rate of two heroes on the same or on opposing teams will be used as input features. However, this must be done with a sufficient amount of historical data. Since the auto battler game is in a fast-growing stage, it needs to be quickly adjusted. The historical win rate data is not of great significance. Further, when adjusting the balance, the model is needed to make a prediction when the new version of the game is not online. Hence, the player's historical data cannot be used to predict the new chess piece. In this study, this prediction model only uses chess features for input.

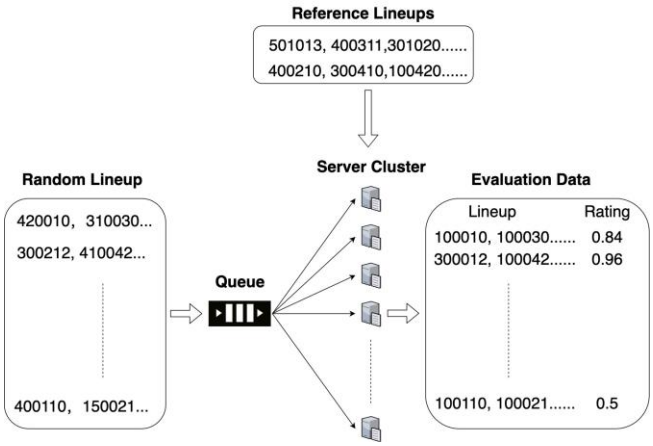

Figure 2. The process of lineup strength calculation

In this study, to comprehensively consider the role of bonus, the same chess piece with a different kind of bonus is defined as a different piece. Under the influence of a different bonus, the strength of the chess game is completely different. Therefore, the possibility of combinations with different star ratings and different bonuses is traversed. In this experimental version of the Chess Rush, there are a total of 2012 possibilities in the game. A vector of 2012 dimensions is defined as the model input. When the state of the chess piece is determined, the corresponding position code is set to one. The schematic diagram of the encoding is shown in Fig. 3. After the above encoding, it can comprehensively reflect the characteristics of the star level and bonus in the chess game, thereby effectively improving the accuracy of the prediction.

The above research on feature selection and encoding of the auto battler game determines the input of the evaluation model. Because this study needs to predict the strength of the lineup, which is a continuous value, it needs to choose a regression model for training and prediction. At present, the commonly used regression model algorithms include a k-

nearest neighbor regression, ridge regression, decision tree regression, SVR regression, and neural networks regression [16]. Different regression algorithms have different advantages, disadvantages, and use range. In this auto battler lineup evaluation model, the input eigenvector dimension is higher, which is the 2102 dimension vector. The overall training data is large, the win rate of nearly 500000 sets of lineups must be learned, which is difficult to be stored in the memory at one time. There is also a certain synergy relationship between pieces, hence the model is expected to learn the inner interaction from the training data.

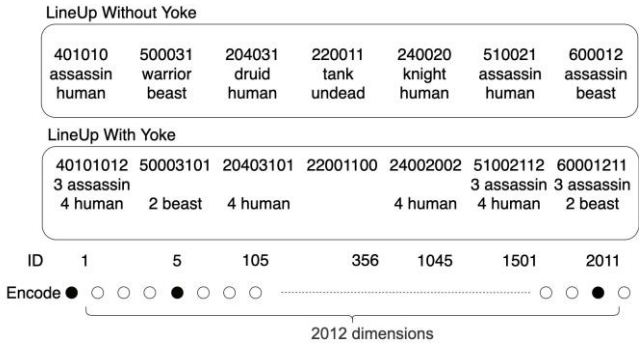

Figure 3. The lineup encode of model

Linear regression is a commonly applied regression algorithm with an overall relatively fast modeling speed in the case of a large amount of data and no complicated calculations [17, 18]. The influence of each feature can be explained according to the coefficients, which is convenient for humans to understand the interior of the model. This has a good effect on small amounts of data and simple relationships. However, this algorithm is more sensitive to outliers, which makes it difficult to fit. Therefore, it is difficult for the LR model to obtain a better training effect for the auto battler evaluation model. For the decision tree method, the algorithm is relatively simple to calculate, easy to understand, and highly interpretable. In particular, it is suitable for processing data with missing samples. However, its overall operation speed is considerably slow, and the memory consumption is excessively high.

Based on the advantages and disadvantages of the above regression model, a neural network model is chosen for this lineup evaluation. The multilayer neural network model has multiple layers of non-linear structure and can express the non-linear relationship between highly complex features. The layer structure of the neural network model can be easily adjusted, and the input and output can likewise be adjusted in a flexible manner. Related research shows that as the amount of training data increases, the overall network model performance becomes better. However, the overall neural network model is more complex, hence it is difficult to interpret and requires powerful equipment for training. In this experiment, the computer resources and training data size meet the needs of the multi-layer neural network regression algorithm. Advantages regarding the data size can be fully utilized for accurate predictions.

B. Lineup Mining Based on Genetic Algorithm

Traditional chess and card games often use the Monte Carlo search tree algorithm to optimize search results. MCTS is a general term for a class of tree search algorithms that can effectively solve some problems with large exploration space. For example, Go algorithms are generally implemented based on MCTS. However, this is aimed at the optimization of the space for two people to take turns in the decision making. In this task, the strong lineup is directly chosen without a draw, hence direct use of the MCTS method is inadequate. Obtaining the optimal lineup training data is cumbersome, which complicates the use a neural network algorithm to generate the strongest lineup. Further, lineup generation needs to be performed with constraints, such as different numbers of chess and cost. Current neural networks have difficulty completing such tasks. The genetic algorithm can quickly complete lineup mining without training data and with multiple constraints. In this study, genetic algorithms are used to investigate the lineup mining of the auto battler.

The genetic algorithm (GA) is a computational model that simulates the biological evolutionary process of natural selection and the genetic mechanism of Darwin's theory of biological evolution [19-21]. It searches the optimal solution by simulating the natural evolutionary process. Its main characteristics involve direct operation on structural objects, and there are no restrictions on the derivative and function continuity. Moreover, it has inherent implicit parallelism and good global optimization capabilities, adopts a probabilistic optimization method that does not require certain rules, and it can automatically obtain and guide the optimized search space to adaptively adjust the search direction. The GA targets all individuals in a population and uses randomization techniques to guide an efficient search of encoded parameter space. Among them, selection, crossover, and mutation constitute the genetic operation of the GA, whereas the five elements of parameter coding, initial population setting, fitness function design, genetic operation design, and control parameter setting constitute the core content of the GA, as shown in Fig. 4.

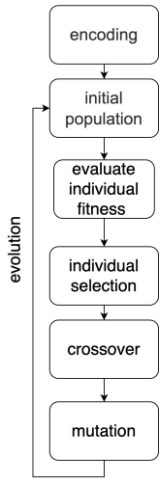

Figure 4. The evolutionary process of genetic algorithm

In the auto battler lineup mining, this study makes targeted adjustments and optimizations to the GA to obtain the required results. Above, one-hot encoding is employed, which is related to the bonus. However, for genetic algorithms, the feasibility of bonuses is not directly related to them, hence the one-hot encoding is directly based on chess pieces. In Chess Rush, there are 162 chess pieces in total, hence the length of the gene in the GA is coded as 162 bits. When a chess piece is selected, the corresponding bit is coded as 1, otherwise, it is 0.

In the traditional GA, the initial population setting is randomly generated according to a certain probability. However, for the auto chess lineup mining, a purely randomly generated initial population will lead to a slower mining speed, and ultimately to a strong target lineup under constrained conditions. The limit of the chess piece is determined before line mining. If the constraint of the number of pieces is 8, the initial gene selects 8 positions from 162 codes. This method reduces the number of iterations and time from the unused lineup over piece number constraints.

The lineup evaluation network studied above is the core of the fitness evaluation function of genetic algorithms. If the lineup is stronger, the lineup's adaptability is considered stronger, such that it has a greater selection opportunity for crossover and mutation in the evolution. After iterating through evolution in this manner, an approximate optimal solution is finally obtained. During the evolution process, there are certain constraints on the lineup that is mined, such as the pieces number or the total cost as in (1).

$$\max \ f_{\text{fitness}}(\text{lineup})$$
$$\mathbf{s.t.}\begin{cases} \text{pieces num of lineup} \le N \\ \text{cost of lineup} \le C \\ \text{three-star pieces num} \le N_{\text{three-star}} \end{cases} \tag{1}$$

If a lineup does not meet the constraints, its fitness function is set to 0.01 to strengthen the individual's constraint and retain a particular population diversity as in (2).

$$f_{\text{fitness}}(\text{lineup}) = \begin{cases} f_{\text{nn}}(\text{lineup}) & \text{satisfy constraints} \\ 0.01 & \text{not satisfy constraints} \end{cases} \tag{2}$$

After the initial population is generated, individuals are selected according to their fitness as in \eqref{eq2}, and crossover and mutation are performed based on the gene.

$$P(\text{lineup } i \text{ reproduces}) = \frac{f(l_i)}{\sum_{k=1}^{n} f(l_k)} \tag{3}$$

Crossover refers to the exchange of genetic coding between two individuals, while mutation means that the coding of a single individual changes randomly. Because of the characteristics of auto battler lineup mining, its genetic individuals need to meet the constraint of the total number of chess pieces. When crossing, two individuals need to choose a suitable position in the gene to exchange, such that the number of chess pieces does not change after crossing. When the mutation takes the positions coded as 1 and 0 from a single gene and reverses it, it is also needed to achieve the constant number of pieces. The control parameters of the GA need to be optimized according to the experimental results, to obtain a balance between the efficiency and results.

IV. Exploratory Analysis

A. A. Lineup Evaluation Model

Our lineup evaluation network adopts the full connection neural network. Because of the large amount of data, it cannot be read into memory at one time. A total of 128 samples are taken from the win rate database each time as a batch of input to the network model for training. After ten epochs, the training is completed. There are three hidden layers in the network. The first two layers comprise 3000 neurons, and the last layer has 1000 neurons. Each hidden layer uses the ReLU activation function, which can make the network training faster. Compared with the Sigmoid and Tanh activation function, it is easier to find the reciprocal. Furthermore, this can increase the nonlinearity of the model and make the network sparse. Because the output range of the final output layer is 0–1, the sigmoid activation function is used in the last layer, which is monotonous and continuous. This function is not easy to diverge in the transfer process. The loss function depicts the mean square error, which is the mean value of the sum of squares of the corresponding point errors of the prediction and the original data.

When there is only one hidden layer, the average absolute error of the network after five iterations is 1.779×10^{-2}. When there are three hidden layers, the average absolute error after five iterations is 1.764×10^{-2}. However, with the further increase in the number of hidden layers to five layers, the average absolute error after training is 1.741×10^{-2}, as shown in Tabel I. A larger number of layers leads to a stronger ability of the fitting function in theory, and thus to a better effect. The deeper layers may lead to overfitting problems, making the model difficult to converge. Hence, a three hidden layer structure is chosen in this experiment.

The linear regression model meets the requirements of this lineup evaluation, which is equivalent to a neural network without a hidden layer. After training, the MAE result is 2.746×10^{-2} of the LR model, which is higher than for a fully connected neural network with hidden layers. However, the advantage of regression models is that they are interpretable. Users can determine the effect of each piece on the final win rate. When the weight of the chess piece is larger, it demonstrates that the chess piece has a greater influence on the win rate, such that a linear regression model can be used as required.

TABLE I. MAE OF DIFFERENT MODEL($\times10^{-2}$)

LR	1 Hidden Layer	3 Hidden Layers	5 Hidden Layers
2.746	1.779	1.764	1.741

We extracted one thousand untrained lineups and calculated the actual win rate of the match and the one obtained by the evaluation model. The data is plotted in the Fig. 5. The abscissa is the true win rate, and the ordinate is

the predicted win rate. The ideal predicted result should be a 45° slash. However, due to the prediction error, the actual points are placed on both sides of the 45° slash. Nevertheless, all points are closer to the ideal value, which proves to have a better prediction effect. Among them, the prediction result close to 1 and 0 is the best one, and the middle value has a poor prediction. The upper and lower slashes in the figure indicate the range of the prediction deviation as ± 0.06. About 97\% of the error of the predicted value is within the range of 0.06. The prediction accuracy of the win rate meets the requirements of the next step of GA mining.

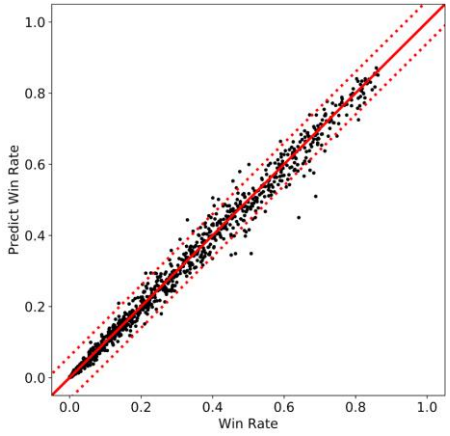

Figure 5. The distribution of predict and true win rate

B. *Lineup Mining Analysis*

In general, the GA requires multiple iterations. As the number of iterations increases, it can gradually converge to the regional optimal solution; however, it will oscillate as it approaches the optimal solution. The Fig. 6 depicts the relationship between the number of iterations and the lineup strength. After the GA reaches the 100th generation, the lineup strength becomes stable. The overall win rate remains unchanged with further iteration rounds.

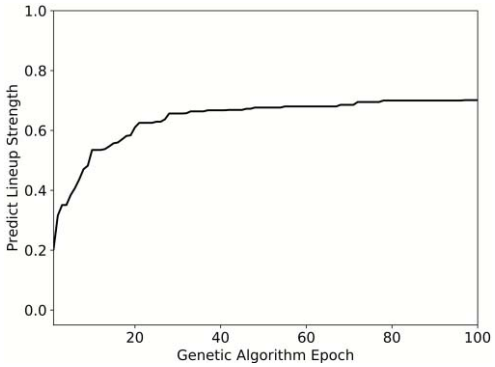

Figure 6. The lineup stength with Genetic Algorithm evolution

The constraints of the lineup need to be determined when using the GA to mine the lineup, because when players draw and combine the lineup, the possibility of different chess games is different, and the total cost is also limited. According to the game developer, four and five coin cost chess pieces are limited to two stars. There are no more than two three-star chess pieces in one lineup, and the first three-star chess piece must be the lowest coin chess piece. Only when the above constraints are met, the mined lineup becomes consistent with the real conditions of players. After the lineup mining with constraints, the win rate comparison of the historical and mined lineups is as follows. The lineup strength comparison of the historical lineup and the mined lineup when the number of pieces is limited to ten, as shown in Fig. 7. The chart shows that the highest lineup strength increases with increasing cost. The red line means the average strength of top 300 usage lineups. The cyan line means the top 5% lineup strength of top 300 usage lineups. The green and blue line represent GA mined lineup with and without traversal. When the cost is 50, the mined lineup strength with traversal is 0.415. When the cost increases to 100, the mined lineup strength with traversal reaches 0.8545. When the cost is in the range of 50–80, the mined lineup with bonus traversal has an obvious advantage in the face of the highest win rate of the historical lineup, which is about 10% higher than the historical lineup on average. This result demonstrates that the mined lineup has a strong advantage, and the lineups are never shown in history. This algorithm can effectively help game developers optimize the balance. However, with the increasing cost, the advantage of the algorithm is not obvious, whereas it nevertheless remains stronger than 95% of the historical lineup.

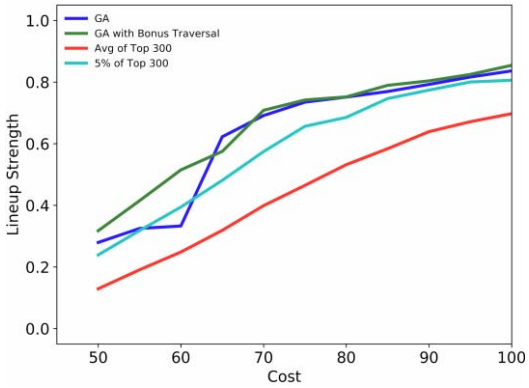

Figure 7. The strongest lineup under different cost

However, the above algorithm has a concern, namely, the obtained results will have certain randomness. After multiple evolutions, they will tend to a single bonus. In fact, the obtained results may be a local best. If GA calculations can be performed for each kind of class and race, this will yield a better overall result. Therefore, we use distributed computing for bonus traversal. Each computer is responsible for the calculation under a single class or race constraint. When the obtained lineup does not satisfy the constraint, the fitness function is set to zero. After multiple rounds of evolutionary calculations, the results of each computer are summarized, as shown in Fig. 8. Thus, a lineup that tends to the global optimal solution can be obtained. The overall strength of the bonus traversal algorithm using distributed computing is higher than the algorithm without traversal.

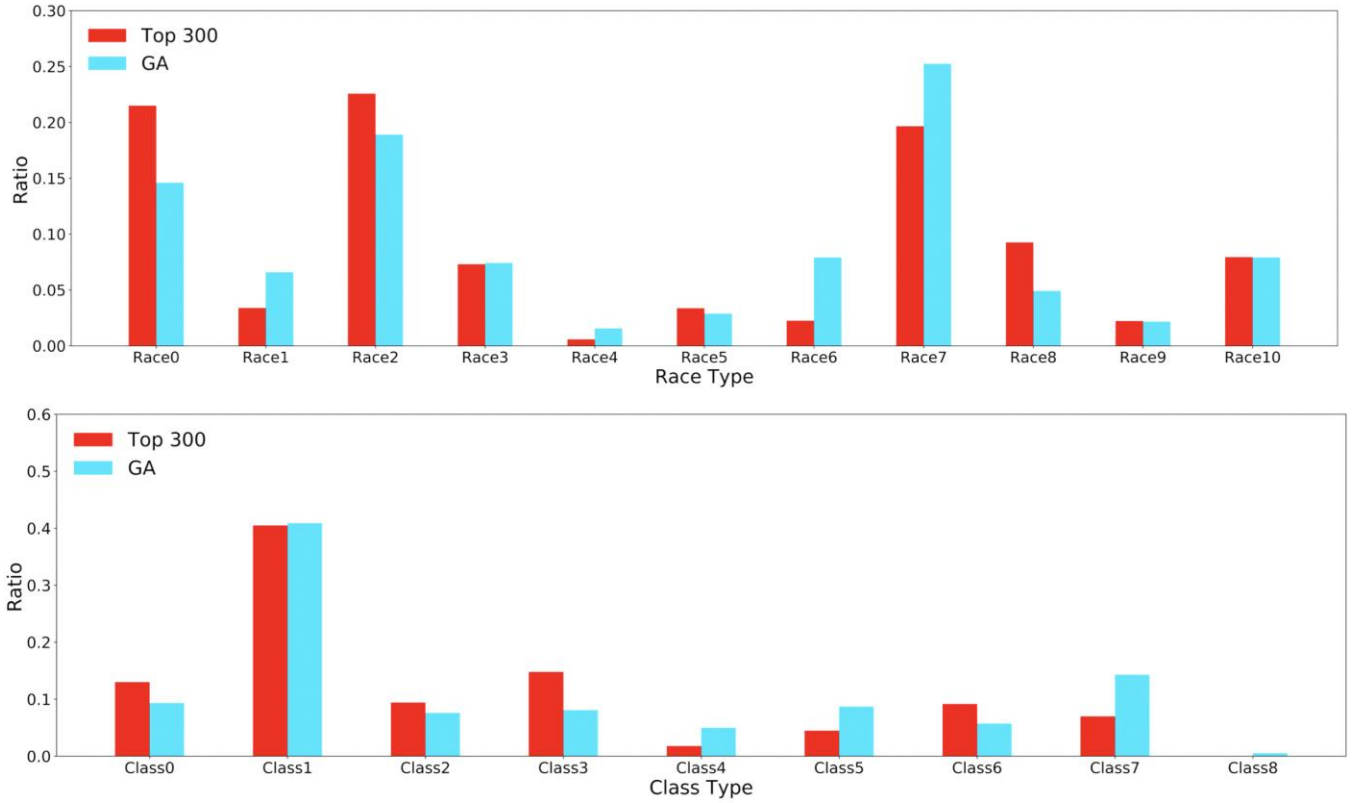

Figure 8. The usage rate of race and class type(because of the need for game confidentiality, the bonuses type are encoded)

V. CONCLUSIONS

This study proposes an evaluation framework for the auto battler, aiming at the needs of a balanced analysis of the auto battler game genre. The reference lineups are constructed as a benchmark, and the win rate of a single lineup against a reference lineup is used as the lineup strength. A large number of lineup strengths are calculated and input to the multilayer neural network model for training. A lineup evaluation model is obtained. Using the results of the lineup evaluation model as a fitness function, the GA was optimized to accelerate its convergence speed. Finally, the lineup mining results are obtained under multiple constraints. The mined lineup outperforms than the existing historical lineup. Game developers can use the obtained lineup to optimize game chess pieces and bonuses and further adjust the overall balance before the game is released.

ACKNOWLEDGMENT

This work is supported by Key-Area Research and Development Program of Guangdong Province (2019B010155003), Shenzhen Science and Technology Innovation Commission (JCYJ20200109114835623), National Natural Science Foundation of China (U1713203), and the Scientific Instrument Developing project of the Chinese Academy of Sciences (YJKYYQ20190028).

REFERENCES

[1] M. Aung, V. Bonometti, A. Drachen, et al. “Predicting skill learning in a large, longitudinal moba dataset,” IEEE Conference on Computational Intelligence and Games, 2017.

[2] A. Katona, R. Spick, V. J. Hodge, et al. “Time to die: death prediction in dota 2 using deep learning,” IEEE Conference on Games, 2019.

[3] Y. Yang, Qin T, Y. H. Lei. “Real-time esports match result prediction,” arXiv preprint arXiv:1701.03162, 2016.

[4] K. Kalyanaraman. “To win or not to win? A prediction model to determine the outcome of a DotA2 match,” Technical report, University of California San Diego, 2014.

[5] N. Kinkade, L. Jolla, and K. Lim. "Dota 2 win prediction." Univ. California, Tech. Rep. 2015.

[6] V. Hodge, S. Devlin, N. Sephton, et al. “Win prediction in esports: Mixed-rank match prediction in multi-player online battle arena games,” arXiv preprint arXiv:1711.06498, 2017.

[7] K. Wang, W. Shang. “Outcome prediction of DOTA2 based on Naïve Bayes classifier,” 16th International Conference on Computer and Information Science, pp.591-593, 2017.

[8] C. E. Almeida, R. C. Correia, D. M. Eler, et al. “Prediction of winners in MOBA games,” 12th Iberian Conference on Information Systems and Technologies, pp. 1-6, 2017.

[9] I. Porokhnenko, P. Polezhaev, A. Shukhman. “Machine Learning Approaches to Choose Heroes in Dota 2,” 24th Conference of Open Innovations Association, pp. 345-350, 2019.

[10] F. Y. Wang, J. J. Zhang, X. Zheng, et al. “Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond,” IEEE/CAA Journal of Automatica Sinica, vol. 3(2), pp. 113-120, 2016.

[11] C. F. Sironi, M. H. M. Winands. “Analysis of self-adaptive monte carlo tree search in general video game playing,” IEEE Conference on Computational Intelligence and Games, 2018.

[12] M. Świechowski, T. Tajmajer, A. Janusz. “Improving hearthstone ai by combining mcts and supervised learning algorithms,” IEEE Conference on Computational Intelligence and Games, 2018.

[13] I. Bravi, D. Perez-Liebana, S. M. Lucas, et al. “Shallow decision-making analysis in general video game playing,” IEEE Conference on Computational Intelligence and Games, 2018.

[14] Q. Jiang, K. Li, B. Du, et al. “DeltaDou: expert-level doudizhu AI through self-play,” Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp.1265-1271, 2019.

[15] L. Zhang, H. Pan, Q. Fan, et al. “GBDT, LR \& Deep Learning for turn-based strategy game AI,” IEEE Conference on Games, 2019.

[16] C. G. Diaz, P. Perry, R. Fiebrink. “Interactive machine learning for more expressive game interactions,” IEEE Conference on Games, 2019.

[17] P. Yang, D. L. Roberts. “Knowledge discovery for characterizing team success or failure in (A) RTS games,” IEEE Conference on Computational Inteligence in Games, 2013.

[18] S. F. Gudmundsson, P. Eisen, E. Poromaa, et al. “Human-like playtesting with deep learning,” IEEE Conference on Computational Intelligence and Games, 2018.

[19] M. J. Kim, C. W. Ahn. “Hybrid fighting game AI using a genetic algorithm and Monte Carlo tree search,” Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp.129-130, 2018.

[20] J. J. Nielsen, M. Scirea. “Balanced map generation using genetic algorithms in the siphon board-game,” International Conference in Software Engineering for Defence Applications, pp. 221-231, 2018.

[21] M. Morosan, R. Poli. “Speeding up genetic algorithm-based game balancing using fitness predictors,” Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp.91-92, 2017.