Real-time Strategy Game Tactical Recommendation Based on Bayesian Network

To cite this article: Yang Zhen et al 2019 J. Phys.: Conf. Ser. 1168 032018

View the article online for updates and enhancements.
Real-time Strategy Game Tactical Recommendation Based on Bayesian Network

Yang Zhen¹, Zhang Wanpeng and Liu Hongfu
National University of Defense Technology, Changsha, China
¹ Corresponding author: yangzhen7319@163.com

Abstract. Real-time strategy (RTS) games simulate battles between large numbers of units and pose significant challenges to artificial intelligence which are complex areas of confrontation. In the field of RTS games, confrontation planning under uncertainty remains an unsolved challenge. There are two main types of uncertainty in RTS games. The first is the partial observability of the game, causing uncertainty. Second, there is uncertainty as the game is against the game and the player cannot predict what the opponent will do. This paper uses the Bayesian model as a logical alternative to solve the uncertainty caused by the confrontation game. The experimental results show that the proposed model can effectively solve the tactical recommendation problem in real-time strategy games.

1. INTRODUCTION

RTS games can be seen as simulating real and complex dynamic environments in a limited and small world, posing an important challenge to the development of artificial intelligence. In recent years, many researchers have used RTS games as test platforms in case-based reasoning and planning, machine learning [1], deep learning [2], and heuristic and confrontational search [3]. To study challenging issues, in the tactical decision-making of RTS games, the acquired opponent information and environmental information are incomplete. Bayesian model is one of the main techniques for the representation of uncertain knowledge in artificial intelligence. It is an excellent tool for dealing with random problems. It can describe uncertain decision tasks well and can clearly display various variables in decision making. The intricate interrelationship has the advantages of being intuitive and easy to understand. At the same time, its mature reasoning technology makes the calculation decision result relatively simple, and can fully exert its advantages and serve as a tactical recommendation service for real-time strategy games.

The application of Bayesian networks in games has attracted the attention of different researchers. Parra [4] proposed a method that can imitate the player's decision as a means of managing micro-fights in the RTS game "StarCraft." The Bayesian network is generated to accommodate the decisions made by the player, and then information from the player's battle micromanagement is collected for training. Finally, the game network is implemented in the game to improve the performance of the built-in artificial intelligence module. This method effectively demonstrates that Bayesian networks can mimic humans to make decisions. Bayesian networks provide a stable, understandable way to generate decision simulations. Hostetler et al. [5] proposed a dynamic Bayesian network strategy model in the RTS game "StarCraft", which can infer the unobserved environmental information from the real observation.
The article uses Bayesian programming to formalize the Bayesian network. Since this form is based only on inference rules that require probability calculus, it is very versatile. This paper proposes a tactical recommendation method used in RTS games to formally model the environmental information and unit information of the game. The experimental results show that the proposed method can effectively solve the tactical recommendation in RTS games.

The remain of the paper is outlined as follows. The second part is an introduction to basic knowledge, including an introduction to RTS games and Bayesian theory. The third is the Bayesian tactical recommendation model in the real-time strategy game proposed in this paper. The fourth part is the experiment to verify the validity and strength of the model. The fifth part is the conclusion.

2. BASIC KNOWLEDGE

2.1 RTS games
The real-time strategy game is essentially a simplified military simulation. RTS games involve long-term goals and often require multiple levels of abstraction and reasoning. Even with the current active field of artificial intelligence research, Go, the complexity of RTS games has increased dramatically. The state space is so large that traditional heuristic-based search techniques have so far failed to solve all the problems except the most restricted sub-problems of RTS AI. Meta-group \( G = (P, S, A, L_u, L_p, T, W, s_{in}) \) is generally used to define real-time strategy games, here [6]:

- \( P = \{ \text{max}, \text{min} \} \), indicating the different players set up.
- \( S \) indicates a possible status. For example, \( \text{units}(p, s) \) indicates a combat unit that belongs to player \( P \) in state \( S \).
- \( A \), which represents a limited set of unit actions that the combat unit can perform.
- \( L_u(u, a, s) \rightarrow \{ \text{true}, \text{false} \} \), which represents a function, returns whether unit \( u \) can perform unit actions \( a \) in state \( S \). For simplicity, \( L_u(u, s) = \{a \in A | L_u(u, a, s) = \text{true} \} \) and \( \text{ready}(p, s) = \{u \in \text{units}(p, s) | L_u(u, s) \neq \emptyset \} \) can also be used to represent this function.
- \( L_p(p, \alpha, s) \rightarrow \{ \text{true}, \text{false} \} \), which represents a function, returns whether player \( P \) can perform player action \( \alpha \) in state \( S \). Given a set of combat units \( \text{ready}(p, s) = \{u_1, ..., u_n\} \), a player's action \( \alpha \) is defined as \( \text{alpha} = \{(u_1, a_1), ..., (u_n, a_n)\} \), making \( L_u(u_i, a_i, p) = \text{true} \) in \( 1 \leq i \leq n \). Thus, the \( \text{ready} \) function determines the set of elements that can perform unit actions, the \( L_u \) function determines which actions each unit can perform, which determines the set of possible player actions, and \( L_p \) determines which possible player actions are legitimate.
- \( T(s, \alpha_{\text{min}}, \alpha_{\text{max}}) \rightarrow S \), which is a deterministic transfer function, gives the state \( s_t \in S \) and player actions \((\alpha_{\text{min}}, \alpha_{\text{min}}, \alpha_{\text{max}})\) for each player at time \( t \), returning the state that will arrive at time \( t+1 \) (ie, \( T \) is the forward model of the game).
- \( W: S \rightarrow \{ \text{max wins}, \text{min wins}, \text{draw}, \text{ongoing} \} \) is a function that determines the winner of the game. The result can be that the game is running, one wins or draws.
- \( s_{\text{inf}} \in S \), indicating a limited state.

2.2 Bayesian network
The Bayesian network, also known as the Belief Network, or the directed acyclic graphical model, is a probability graph model. First proposed by Judea Pearl [7] in 1985, it is a model of uncertainty processing that simulates causality in human reasoning. The Bayesian network is formed by plotting the random variables involved in a research system according to whether the conditions are independently drawn in a directed graph. It is mainly used to describe the conditional dependencies between random variables, using circles to represent random variables and arrows to indicate
conditional dependencies. The nodes in the directed acyclic graph (DAG) of the Bayesian network represent random variables $\{X_1, X_2, \ldots, X_n\}$, which can be observable variables, or hidden variables, unknown parameters, and so on. Variables or propositions that are considered causal (or unconditionally independent) are connected by arrows (in other words, the arrows connecting the two nodes represent whether the two random variables are causal or unconditionally independent). If two nodes are connected by a single arrow, indicating that one of the nodes is "parents" and the other is "children", the two nodes will generate a conditional probability value.

Let $G = (I, E)$ denote a DAG, which represents a collection of all nodes in the graph, and represents a set of directed connected segments, and has $X = \{X_i\}_{i \in I}$ represented by a node in its directed acyclic graph. Random variable, if the joint probability of node $X$ can be expressed as:

$$p(x) = \prod_{a \in I} p(x_i | x_{pa(i)})$$  \hspace{1cm} (1)

In addition, for any random variable, the joint probability can be obtained by multiplying the respective local conditional probability distributions:

$$p(x_1, \ldots, x_n) = p(x_1 | x_2, \ldots, x_n) \ldots p(x_n)$$ \hspace{1cm} (2)

The Bayesian network combines graph theory and statistical knowledge to provide a natural representation of causal information, along with other knowledge discovery methods and decision modelling methods (e.g. rule representation, decision trees, artificial neural networks, etc.) Compared with the following advantages:

- Bayesian networks can unearth the implications of knowledge. After learning the Bayesian network from the data, the network is reasoned, interpreted, and able to obtain the desired knowledge, concepts, and decision information. Moreover, from the network model after learning, explicit knowledge can be extracted, and black box defects such as neural network models can be avoided. The conditional probability can express the correlation between various information elements, which can be limited and incomplete. Learning and reasoning under uncertain information conditions.

- The Bayesian network itself is an uncertain causal correlation model. Different from other decision models, Bayesian network itself is a kind of probabilistic knowledge representation and reasoning model that visualizes multivariate knowledge schemas. It more closely implies the causal relationship and conditional correlation between network node variables, which is convenient for analysing action sequences. The result of the action, the interaction with the observation, and the expected effect of the action, so that planning and decision making can be made under uncertainty.

- Bayesian networks have parallel reasoning capabilities and global update capabilities. The Bayesian network takes values based on the node variables in the network. Over-probability reasoning can obtain the posterior probability information of any other node variable to achieve global update.

- The simple expression and conditional independence of Bayesian network saves storage space, simplifies the process of knowledge acquisition and domain modelling, and reduces the complexity of the reasoning process. At the same time, graphical representation makes efficient reasoning possible, using distributed belief update. The method greatly improves the computational efficiency and avoids the repeated interaction of causal relationships.

2.3 Bayesian programming

Bayesian programming is used to formalize the Bayesian network. Since this form is based only on inference rules that require probability calculus, it is very versatile. Bayesian programming is a formal method used to fully describe the Bayesian model, which includes Bayesian networks and Bayesian mapping. In fact, it is equivalent to the probability factor graph. There are two main types of problems in the Bayesian network. One is how to describe and calculate the joint release, and the second is how to solve the problem. The description includes showing the associated variable $X^\pi$ and data $\delta$ to solve the joint distribution to explain the dependencies between them. The form of the Bayesian model shows how to calculate their distribution. Formal
representations can be parameterized or recursive to other Bayesian programming. Conditional independence is represented in the decomposition $P(\text{Searched}, \text{Free}, \text{Known})$. \text{Searched} and \text{Known} are two disjoint subsets of variables used to calculate the distribution.

$$P(\text{Searched} | \text{Known}) = \sum_{\text{Free}} P(\text{Searched}, \text{Free}, \text{Known})$$
$$= \frac{1}{Z} \sum_{\text{Free}} P(\text{Searched}, \text{Free}, \text{Known})$$

(3)

The complete model is summarized by Bayesian programming, expressed as:

$$P(\text{Searched} | \text{Known}) = \frac{1}{Z} \sum_{\text{Free}} P(\text{Searched}, \text{Free}, \text{Known})$$

3. TACTICAL RECOMMENDATION MODEL

Hagelbäck and Johansson [8] found that "tactics are one of the most successful indicators of whether a player is a human being" when studying the intelligent features of RTS games. The tactics are between strategic (advanced) and micro-management (lower), covering where the attack is and how to attack. A good human tactical decision maker needs to consider a lot of questions when choosing tactics: Is there a flaw in defense? Which place is worthier of attack? What is the number of units I am attacking here? Is the terrain (the fortress point) good for me? and many more. The problem that this article needs to solve is to consider as much as possible the tactical problems in real-time strategy game confrontation planning and make the most effective intelligent recommendation. The established intelligent recommendation system performs the function of "someone" to direct the collaboration of the unmanned platform. Different units have different abilities, which leads to different tactics possible. Consider three different combat units here. Light combat unit (marine), heavy combat unit (firebat), remote combat unit (ghost). Attacking enemy bases is an important tactical action that directly affects the course of operations.

In the intelligent recommendation of tactics, it is first necessary to predict the tactics of the opponents. On this basis, the intelligent recommendation in the tactical game process is carried out to optimize the game results. That is, taking into account all the things that can happen, make the most effective tactical decisions and recommendations. The complexity of tactics has not been specifically studied in the existing literature. The tactics correspond to where the player moves a group of combat units and how they move and move. Combat units have different capabilities, which leads to different possible tactics. For attack tactics, Bayesian programming has the following definitions:

1. **Variables:** Combat units are distributed in $n$ areas

   (1) $A_i \in \{\text{true, false}\}, A_i$ indicates whether the opponent is attacked in the area $i$. \text{true} indicates that the opponent is attacked in the area, and \text{false} indicates that the opponent is not attacked in the area.

   (2) $E_i \in \{\text{no, high}\}, E_i$ is the economic value of the defender in the area $i$. That is, if there is a base for defenders in the area, the economic value of the area is \text{high}; in the area where there is no base for defenders, the economic value of the area is \text{no}.

   (3) $T_i \in \text{tactical\_score}(r), T_i$ is the tactical value of the defender in the area $i$, and is a discretized form of data. Represents the ratio of the sum of the value of all defense units in the area $i$ to the sum of the values of all regional defense units. Formal representation is:

   $$\text{tactical\_score}(r) = \frac{\sum_{i \in \text{region}} v^d_{\text{type}}(r)}{\sum_{i \in \text{region}} v^d_{\text{type}}(i)}$$

   (5)

   $v^d_{\text{type}}(r)$ represents the sum of the values of the defense units($d$) for all given types of (type) in the area $i$. 


(4) \( T_{A,i} \in \text{tactical}_i \text{score}^T(r) \), \( T_A \) is the tactical value of the attacker in the area \( i \), which is the same as the definition of the above defender.

(5) \( H_{i,0} \in \{ \text{marine, firebat, ghost} \} \), \( H_i \) indicates the attack mode used in the area. These three represent three different combat units. Different attack units form different combat tactics.

(6) \( MD_{i,n} \in \text{marine}_i \text{defense}^d(r) \), \( MD \) indicates the ratio of the sum of the value of the attacker's marine unit that can attack the attacker's marine unit to the sum of all the unit values of the attacker in the area \( i \). The formal representation is as follows:

\[
\text{marine}_i \text{defense}^d(r) = \frac{V_{\text{marine} \text{units}}^d}{V_{\text{marine} \text{units}}^a} (6)
\]

(7) \( FD_{i,n} \in \text{firebat}_i \text{defense}^d (r) \), \( FD \) indicates the ratio of the sum of the value of the attacker's firebat units that can attack the attacker's firebat units to the sum of all units values of the attacker in the area \( i \). The formal representation is the same as \( MD \).

(9) \( GD_{i,n} \in \text{ghost}_i \text{defense}^d (r) \), \( GD \) indicates the ratio of the sum of the value of the attacker's ghost unit that can attack the attacker's ghost unit to the sum of all the unit values of the attacker in the area \( i \). The formal representation is as follows:

\[
\text{ghost}_i \text{defense}^d (r) = \frac{V_{\text{ghost} \text{units}}^d}{V_{\text{ghost} \text{units}}^a} (10)
\]

All possible tactical attack combinations for different combat units. A reasonable probability distribution table can be established by real-time collecting the cell information existing in the environment.

2. Decomposition: Combat units are distributed in \( n \) areas

\[
P(A, E, T, H, MD, FD, GD, HP) = \prod_{i=1}^{n} P(A_i)P(E_i)P(T_i)P(H_i|A_i)P(MD_i|E_i,H_i)P(FD_i|E_i,H_i)P(GD_i|E_i,H_i,HP_i)P(HP_i) (7)
\]

The Bayesian network tactical model is shown in the figure.

![Bayesian Network](image)

Figure 1. Attack tactics Bayesian network

3. Forms and Learning

Formal representation and parameter learning for each part of the joint distribution. For a given area, there is the following representation.

(1) A prior probability that the player is performing an attack in this area. In this model, the proportion between the combat unit and the total unit that performs the attack in this area is set to \( P(A) = \frac{n_{\text{battles}}}{n_{\text{battles}} + n_{\text{not battles}}} \). In a given learning data, it should be uniformly initialized to gradually learn the opponent's preferred attack area to predict the possibility that an opponent might attack the area in the future, and learn the areas where our attack failed or made a successful decision.

(2) \( P(E, T, TA|A) \) is a joint probability distribution table of economics and tactics (defenders and attackers), indicating the score at the time of the attack. The Laplace's law of succession is used to estimate the probability estimate from the training set, and the maximum likelihood learning is performed on the probability distribution table.
\[ P(E=e, T = t, TA = ta \mid A = \text{true}) = \frac{1 + n_{\text{battles}}(e, t, ta)}{|E \parallel T \parallel TA| + \sum_{E, T, TA} n_{\text{battles}}(E, T, TA)} \quad (8) \]

(3) \( P(MD, FD, GD \mid H) \) is a joint probability distribution table of different attack tactical combinations when an attack occurs. Use Laplace's law of inheritance to learn the data set and populate the probability table.

\[ P(MD = md, FD = fd, GD = gd \mid H = h) = \frac{1 + n_{\text{battles}}(md, fd, gd, h)}{|MD \parallel FD \parallel GD| + \sum_{MD, FD, GD} n_{\text{battles}}(MD, FD, GD, h)} \quad (9) \]

(4) \( P(H \mid HP) \) is how the attack occurring, depending on the possible tactical situation. For example, \( P(H=\text{marine} \mid HP=\text{marine}) = 1.0 \). The general situation is expressed as:

\[ P(H=h \mid HP = hp) = \frac{1 + n_{\text{battles}}(h, hp)}{|H| + \sum_{H} n_{\text{battles}}(H, hp)} \quad (10) \]

(5) \( P(HP) \) is the all possible tactical attack combinations for different combat units. If researches know the constraints of the tactical background, then \( P(HP = hp) = 1 \). Otherwise, researches need to be able to analyse the opponent's tactics for uncertainty, so that researches can obtain a possible distribution \( P(HP) \) for the opponent's tactical attack.

### 4. Questions

For a given area \( i \), the possibility of attacking here is

\[ P(A_i = a_i \mid e_i, t_i, ta_i) = \frac{P(e_i, t_i, ta_i \mid a_i)P(a_i)}{\sum_A P(e_i, t_i, ta_i \mid A)P(A)} \quad (11) \]

The attack tactic should take is

\[ P(H_i = h_i \mid md_i, fd_i, gd_i) = \sum_{hp} [P(md_i, fd_i, gd_i \mid h_i)P(h_i \mid HP)P(HP)] \quad (12) \]

### 4. Experimental analysis and verification

#### 4.1 Experiment setup

This article uses the MicroRTS platform as a model for verification. Developed by Santiago Ontañón [9], MicroRTS is a simple RTS game designed to perform artificial intelligence research while minimizing the amount of work required to participate in the work.
MicroRTS is a two-player Zero-sum Game that demonstrates the standard features of RTS games, especially heterogeneous units with continuous motion that can be performed simultaneously. The environment is fully observable and each grid unit can be occupied by a single unit or by a building or resource store. Each store has a limited supply of resources and they can be used by both players. The figure below shows the basic elements of MicroRTS.

In order to provide data for the training of the model, the researchers first need to run the platform to generate a large amount of playback. This paper selected four hard-coded bots built into MicroRTS (WorkerRush, LightRush, HeavyRush, and RangedRush) as the strategy recommendation method for Player One and Player II, respectively, on eight different maps. In addition, people repeated these rounds three times, giving four hard-coded bots a budget of 500, 1000, and 5000 time frames per game, resulting in twelve data sets:

- $S_{500}^{WR}$, $S_{1000}^{WR}$, $S_{5000}^{WR}$: A data set consisting of all unit actions generated by the WorkerRush hard-coded bots running 500, 1000, and 5000 timeframe games, respectively.
- $S_{500}^{LR}$, $S_{1000}^{LR}$, $S_{5000}^{LR}$: A data set consisting of all unit actions generated by the LightRush hard-coded bots running 500, 1000, and 5000 timeframe games, respectively.
- $S_{500}^{HR}$, $S_{1000}^{HR}$, $S_{5000}^{HR}$: A data set consisting of all unit actions generated by the HeavyRush hard-coded bots running 500, 1000, and 5000 timeframe games, respectively.
- $S_{500}^{RR}$, $S_{1000}^{RR}$, $S_{5000}^{RR}$: A data set consisting of all unit actions generated by the RangedRush hard-coded bots running 500, 1000, and 5000 timeframe games, respectively.

4.2 Experiment analysis

First generate the data set needed to train the model. Used to learn the probability distribution of the model. The two sides used the four hard-coded bots built in MicroRTS to perform round-robin competition on eight different maps, that is, $4 \times 4 \times 8 = 128$ field competitions were performed, and three sets of data were set for 500, 1000 and 5000 time frames respectively. The Bayesian model is trained in the presence of the data set required to train the model. Save the generated model as an XML file locally. The trained data set is divided into twelve categories according to different hard-coded bots and running game frames, and twelve models are respectively trained to compare the influence of different data sets and running game frames on the training model. The twelve models are defined as: $M_{500}^{WR}$, $M_{1000}^{WR}$, $M_{5000}^{WR}$, $M_{500}^{LR}$, $M_{1000}^{LR}$, $M_{5000}^{LR}$, $M_{500}^{HR}$, $M_{1000}^{HR}$, $M_{5000}^{HR}$, $M_{500}^{RR}$, $M_{1000}^{RR}$, $M_{5000}^{RR}$. The paper evaluated the Bayesian probability model proposed above in two different ways. The first way is to measure whether the model accurately predicts the behavior of the built-in intelligent algorithm by cross-checking; the second way is to use the model to test the strength of the model against the same four built-in intelligent algorithms used to generate training data.
4.2.1 Model accuracy

The first is to verify the accuracy of the model. The cross-validation method was used to verify the prediction accuracy of the twelve models. Ten percent of the data samples were extracted from the data set for model testing. The unit motion generated by the model is compared with the data in the test sample to obtain the accuracy of the model prediction.

Table 1. Accuracy test results.

| Mod      | $M_{WR}^{500}$ | $M_{WR}^{1000}$ | $M_{WR}^{5000}$ | $M_{LR}^{500}$ | $M_{LR}^{1000}$ | $M_{LR}^{5000}$ |
|----------|----------------|-----------------|-----------------|----------------|-----------------|-----------------|
| ACC      | 0.650          | 0.654           | 0.656           | 0.878          | 0.879           | 0.881           |

The model is a good predictor of the behaviour of the four hard-coded bots. In particular, the model trained for the data generated by the RangedRush bot is the most accurate. In addition, the higher the time frame calculation budget of the operation, the higher the accuracy of the model prediction. The prediction accuracy of the model has increased, indicating that they may converge to a more stable policy recommendation. However, the accuracy of the increase in budget time frames has not increased significantly, as many of the confrontations have been completed in a limited time.

4.2.2 Model strength

The next step is to verify the strength of the model. The first is to test the strength between the built-in algorithms. In order to find out the impact of the experimental data generated by the built-in algorithm on the different models trained. Secondly, the Bayesian programming model proposed in this chapter and the intelligent algorithm training the model are respectively cyclically confronted on the eight maps of the above training model. Only experimental models with budget time frames of 500 and 1000 are validated here. The benefit of this is to reduce the required runtime based on ensuring the strength of the model verification. In order to more intuitively display the strength of the model proposed in this paper, join the other two built-in intelligent methods Rnd (the square strategy randomly generated by unit motion) and RndBiased (the strategy of randomly generating unit motion, but increase the unit execution attack strategy) The possibility of confrontation. Here, 1 point is won for one game and 0.5 points for each draw. The results of the experiment are shown in the figure below. The data in the table shows the win rate. The first abscissa of the table controls the intelligent algorithm used by player 1. The first column of the table is the intelligent algorithm used by Player 2.

Table 2. Performance of built-in algorithms in MicroRTS.

| Bot | Rd | RdB | WR | LR | HR | RR |
|-----|----|-----|----|----|----|----|
| Rd  | 0.50 | 0.906 | 1  | 1  | 0.938 | 0.938 |
| RB  | 0.025 | 0.50 | 1  | 0.938 | 0.867 | 1   |
| WR  | 0  | 0   | 0.5 | 0  | 0   | 0   |
| LR  | 0  | 0.0625 | 1  | 0.5 | 0.125 | 0   |
| HR  | 0.0625 | 0.1875 | 1  | 0.875 | 0.5  | 0.5  |
| RR  | 0.0625 | 0  | 1  | 1  | 0.5  | 0.5  |
WorkerRush has the highest intensity among the six built-in smart methods used in the test. Mainly because WorkerRush uses a very aggressive offensive strategy that works well against other less aggressive strategies. The next step is to verify the strength of the model. The experimental results are shown in the figure below. The first column of the table controls the intelligent algorithm used by player 1. The first abscissa of the table is the intelligent algorithm used by Player 2.

Table 3. Bayesian model algorithm performance comparison.

| Bot   | Rd  | RdB | WR  | LR  | HR  | RR  | AVG |
|-------|-----|-----|-----|-----|-----|-----|-----|
| $M_{WR}^{500}$ | 1   | 1   | 0   | 0.69 | 1   | 0.94 | 0.77 |
| $M_{WR}^{1000}$ | 0.94 | 0.88 | 0   | 0.94 | 1   | 0.94 | 0.78 |
| $M_{LR}^{500}$ | 0.69 | 0.38 | 0   | 0   | 0   | 0.25 | 0.22 |
| $M_{LR}^{1000}$ | 0.69 | 0.44 | 0   | 0   | 0   | 0.63 | 0.30 |
| $M_{LR}^{500}$ | 0.63 | 0.31 | 0   | 0   | 0   | 0   | 0.15 |
| $M_{LR}^{1000}$ | 0.44 | 0.25 | 0   | 0   | 0   | 0   | 0.12 |
| $M_{RR}^{500}$ | 0.22 | 0.13 | 0   | 0   | 0   | 0   | 0.06 |
| $M_{RR}^{1000}$ | 0.31 | 0.41 | 0   | 0   | 0   | 0.03 | 0.13 |

As can be seen from the above table, the highest score was obtained using the $M_{WR}^{1000}$ model. In summary, when the model training uses the data set generated by the higher computational budget for training, the model can obtain better scores, and the strength of the model is significantly improved. The powerful training method generated by the data generated by the intelligent method has a correspondingly higher intensity.

5. conclusions

This paper studies the machine intelligence-based approach to solving strategic intelligent recommendation techniques in decision making to consider more complex situations. In the strategic intelligence recommendation, the acquired opponent information and environmental information are incomplete. Using the Bayesian model as a logical alternative, this incompleteness is transformed into uncertainty for solving. The experimental results show that the Bayesian programming model proposed in this paper can effectively support the strategy intelligent recommendation technology in decision making, and the accuracy of the model achieve good results. The next step is to add transfer learning to the model to make the model more efficient.

References

[1] Synnaeve G, Bessiere P. A Bayesian Model for Opening Prediction in RTS Games with Application to StarCraft[J]. Proceedings of IEEE Cig Seoul South Korea, 2011, 32(14):281-288.
[2] Foerster J, Nardelli N, Farquhar G, et al. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning[J]. 2017.
[3] Barriga N A, Stanescu M, Buro M. Puppet Search: Enhancing Scripted Behavior by Look-Ahead Search with Applications to Real-Time Strategy Games[C]// Aiide. 2015.
[4] Parra R, Garrido L. Bayesian Networks for Micromanagement Decision Imitation in the RTS Game Starcraft[M]// Advances in Computational Intelligence. Springer Berlin Heidelberg, 2012:433-443.

[5] Hostetler J, Dereszynski E, Dietterich T, et al. Inferring strategies from limited reconnaissance in real-time strategy games[C]/ AUAIPress, 2012

[6] Ontañón S. Combinatorial Multi-armed Bandits for Real-Time Strategy Games[J]. 2017.

[7] Judea Pearl. Fusion, propagation, and structuring in belief networks[J]. Artificial Intelligence, 1986, 29(3):241-288.

[8] Hagelbäck J, Johansson S J. A Multiagent Potential Field-Based Bot for Real-Time Strategy Games[J]. International Journal of Computer Games Technology, 2013, 2009(1).

[9] Ontanón S. The combinatorial multi-armed bandit problem and its application to real-time strategy games[C]/Ninth Artificial Intelligence and Interactive Digital Entertainment Conference. 2013.