A Dynamic Hierarchical Evaluating Network for Real-Time Strategy Games

Weilong Yang[1], *Qi* Zhang[1], *Yong* Peng[1]

[1]Department of Modeling and Simulation, National University of Defense Technology, Changsha, China

Abstract. Researches of AI planning in Real-Time Strategy (RTS) games have been widely applied to human behavior modeling and combat simulation. State evaluation is an important research area for AI planning, which ensures the decision accuracy. Since complex interactions exist among different game aspects, the weighted average model usually cannot be well used to compute the evaluation of game state, which results in misleading player's generation strategy. In this paper, we take dynamic changes and player's preference into consideration, analyze player's preference and units' relationships base on game theory and propose a dynamic hierarchical evaluating network, denoted as DHEN. Experiments show that the modified evaluating algorithm can effectively improve the accuracy of task planning algorithm for RTS games.

1 Introduction

Real-Time Strategy (RTS) games are popular real-time combat simulation games in which players instruct units to gather resources, build structures, destroy opponent's buildings to win the game. As typical agent-based game, RTS games pose a huge challenge for AI researchers due to the large state space, limited decision time and dynamic adversarial environment involved. AI planning becomes an important research area for real-time adversarial planning and non-determination decision [1]. State evaluation is an important part of AI planning. By calculating correlative factors and player's preference, the evaluation of game state is obtained, which can be used to judge whether the state is advantageous for player or not. As basic algorithm for player's planning and decision, evaluation method can influence the decision process, and improve the performance of AI planning in games.

Current evaluating algorithms take related factors into consideration to obtain a state score. This method is applicable to simple and fixed game scene. However, in RTS games, the evaluating factors are constantly changing, for which fixed evaluating algorithm cannot accurately describe the game state at different game phases. For example, the current evaluating function does not consider the spatial relationship between units, resulting in that units at different positions still are calculated by carried resources and hit-point value.

This paper focuses on the evaluating process in RTS games and constructs a hierarchical network similar to HTN planning method. By decomposing evaluating factors, the relationships between different factors are analysed. Considering the changing weights, a dynamic hierarchical evaluating network is constructed and a dynamic weights calculating algorithm is proposed. In the remainder of this paper, after discussing related work, we first present the definition of hierarchical factor network. Then we propose dynamic weight calculating method, followed by extensions to apply it to μRTS games, a minimalistic RTS game used for planning algorithm evaluation.

2 Related Works

2.1. RTS games

RTS games are regarded as a simplification of combat simulation and could therefore serve as a test bed for investigating activities such as real-time adversarial planning and decision making under uncertainty [2]. Compared with conventional board games, RTS games have the following primary differences [3]:

1. Players can pursue actions simultaneously with the actions of other players, and need not take turns.

2. Player actions can be conducted over very short decision times, allowing for rapid sequences of actions. While Player actions are durative, in that an action requires numerous time steps to be executed.

3. The state space and branch factors are typically very large. For example, a typical 128×128 map in "StarCraft" generally includes about 400 player controllable units. Considering only the location of each unit, the number of possible states is about 101685, whereas the state space of chess is typically estimated to be around 1050.

4. The environment of an RTS game is dynamic and non-determinability. The opponent's action is not

controllable and the executing task may be broken down by the uncertain environment.

Research has been conducted to modify game tree search methods to address these differences. Chung investigated the applicability of game tree Monte Carlo simulations in RTS games [4]. Balla and Fern applied the upper confidence bound for trees (UCT) algorithm in an RTS game to address the complications associated with durative actions [5]. Churchill addressed the complications associated with simultaneous and durative actions by extending the alpha–beta search process [6]. Methods such as combinatorial multi-armed bandits have attempted to address the challenge of large branching factors [7-8]. Ontañón and Buro combined the hierarchical task network (HTN) planning approach with game tree search to develop what was denoted as adversarial HTN [9]. Among these planning algorithms, evaluating algorithm is needed to calculate the current and play-out game state.

2.2 State Evaluation in RTS games

Alexander [10] first described evaluating process in RTS game state, proposing an evaluating function by calculating the hit-point and attack ability of units. LTD (Life-Time Damage) is a modified evaluating algorithm, which based on the lifetime damage each unit can inflict. And by weaken the contribution of hit-point, LTD2 algorithm is proposed [9]. Tung [11] combined the LTD method and the unit portfolio evaluation to evaluate game state, but did not consider the relationship between two algorithms.

Taking nonterminal position into consideration, Stanescu [12] modified the state evaluating algorithm by taking other properties into account, such as resources, visibility, security, and goals. By putting forward the concept of visibility and detection, a formula is given to calculate the efficiency of detection. Graham Erickson [13] proposed a global evaluating function model for predicting which side would win after a period of game play-out. Taking military features, economic and player skill into consideration, the logistic regression is used to learn the model weights. Alberto [14] used a contrastive evaluating algorithm to obtain the score by calculating the difference between player and its opponent. Bakkes [15] generated evaluating function by using a central data store of samples of gameplay experiences.

Besides traditional evaluating function method with expert knowledge, learning method is also used to evaluate game state. Li [16] used a GA-based reinforcement learning method named ELM to calculate the optimal unit combination strategy, and building strategy. Marius [17] proposed the use of neural network method for evaluation, the spatial relationships between units were also considered in the category. Learning method can achieve good performance but need pre-learning and large dataset, and if the game parameters changes, it will be useless unless the dataset for changed parameters is provided.

Previous researches mostly use fixed factors and weights to evaluate the state, the relationships between factors are not considered. In this paper, we use hierarchical evaluating network to express factors and relations. By taking game state and player's evaluating principle into consideration, a dynamic weight generation algorithm is proposed, which can help evaluating game state more accurately.

3 Hierarchical Evaluating Networks

Evaluating algorithm is the basis of decision-making method, which can lead the planning process. There are two important aspects in evaluating game state: evaluating factors and factors' weight. This section employs a hierarchical network to manage evaluating factors and use regression algorithm to calculate the dynamic weights.

3.1. Hierarchical factor network

The evaluation of game state reflects the players' judgment of game state by taking different aspects factors into consideration. For different level issues, the evaluating factors are also different. In simple games such as Super Mario, the winning gold is the only factor which reflects state value. For RTS games, considering the complexity and the diversity of game process, the more factors are considered, the more comprehensive the judgment can be.

HTN is an automatic planning method using hierarchical approach to decompose complicated task into sub-tasks until all the sub-tasks can be executed [18]. Since the planning process is similar to human decision process, HTN algorithm achieves satisfying results in solving complex problem and is more sufficient to handle large state space than other planning algorithms [19]. The domain knowledge of HTN planning includes compound actions, atomic actions, and patterns.

Inspiring by HTN process of solving problems, we propose a hierarchical evaluating network for RTS games. Similar to HTN, HEN is a tree structure network, in which nodes can be classified as compound nodes and primitive nodes. Compound nodes represent tasks and missions, which are composed of primitive nodes and compound nodes. Primitive nodes stand for state attributes that can be directly calculated. By analysing game state from different levels and perspectives, the evaluating aspects are systematically considered. For RTS games, HTN networks are mostly divided into two or three layers [20]. Two-layer includes macro layer and micro layer, macro layer represents for player goal and global strategy and micro layer represents for teams' or units' control and actions. Three-layer includes strategy layer, tactical layer and reaction layer. Each layer corresponds to different planning level.

We use the three-layer network to structure the hierarchical evaluating network since it can describe the game state more comprehensive. The strategy layer stands for player's top strategy and winning goal. The tactical layer focuses on the player's tactical strategy, such as opponent modelling, task assignment and development strategy selection. The reaction layer is indicators of tactical task which can be calculated directly, such units' position for tactical position selection, and map's occupied information for path planning. By decomposing factors from strategy layer to reaction layer, a factor tree is constructed. In the tree, upper layer is decomposed into the lower layer.

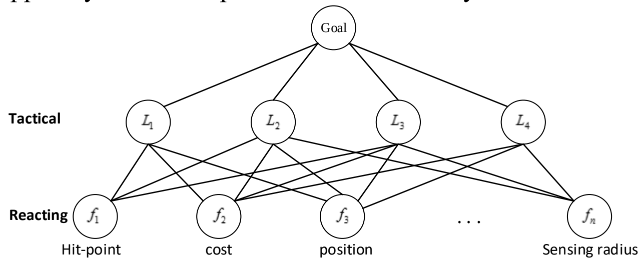

Fig 1. Illustration of a network generated by the HEN algorithm for depth 3.

As shown in Figure 1, in the HEN network, aspects are integrated and all layers can be decomposed to factors. The connection lines between indicators stand for the relationships among the indicators at each level. By constructing network between indicators, it can be found that one sub-indicator can affect multiple upper-level indicators. The evaluation of game state and factor relationships can be expressed as follow:

$$\begin{cases} E(s) = \sum_{i=1}^{n}\sum_{j=1}^{m} E_i(f_j) \\ E(f) = (factor_{father}, factor_{sub}, \text{Attribute}, Weight) \end{cases} \quad (1)$$

The evaluation of game state $E(s)$ is the summation of all aspects' scores. $E_i(f_j)$ stands for the evaluation of each factors. $factor_{father}$ stands for the father factor, $factor_{sub}$ stands for the sub factor. *A*ttribute stands for the attributes of current factor. $Weight$ is the current factor's weight of its father task.

Using hierarchical network to calculate the state situation, a comprehensive attention can be given to both overall and detail factors. In Table 1, the domain knowledge used in DHEN to construct the hierarchical network is listed. The tactical layer includes 4 different aspects, and the reacting layer includes 10 indicators which can be calculated.

Table 1. Domain Knowledge of DHEN.

Tactical	Reacting
Economic Military Building Sensor radius	Resources in units Resource in base Total resource Unit number Work number Light rush number Heavy rush number Building cost Building hit-point Unit position

3.2. Dynamic weight

The essence of RTS game is a two-player zero-sum game. Traditional planning algorithms usually use pre-given factor's weights to evaluate game state, which means the evaluation is fixed from game beginning to the end. But in fact, along with the game evolving, player's preference will change, which means the evaluating principle should change. In this section, we employ three dynamic indexes to describe game dynamic changes: player's strategy S, game time T, and game state $Situation$.

Player's strategy S refers to the initial strategy adopted by each player for gaming. For RTS games, it refers to the game AI used in different players, which designed by hardcode or auto-planning technology. For example, for aggressive decision makers, in the early stages of game, they prefer to create initial forces to attack; for technology decision makers, they prefer to upgrade technology. $G = (\min, \max, p)$ stands for player's goal. Different game principles can be sorted as three types: radical, conservative, balanced. For example, for radical players, they try to destroy his enemy leaving out consideration about their own loss, and their principles can be expressed as $G = \min(b)$. In Table 2 shows the principles of typical RTS AI methods:

Table 2. Typical AI methods.

AI	Strategy
Random	A random strategy AI which executes actions randomly.
Worker Rush	A hardcoded rush strategy that constantly produces workers and sends them to attack.
Light Rush	A rush AI which builds a barracks, and then constantly produces light military units to attack the nearest target (it uses one worker to mine resources).
Heavy Rush	Identical to LightRush, except for producing slower but stronger heavy units.
Monte Carlo	A standard Monte Carlo search algorithm: for each legal player action, it runs as many simulations as possible to estimate their expected reward.
AHTN	An Adversarial Hierarchical Task Network, which combines minimax game tree search with HTN planning.

Game time T affects the evaluating principle in two aspects: one is that since the total game time is given, players' strategies must consider the game time comparing with the final game time. On the other hand, even the same situation may have different effects at different times. Using an exponential decline method, the player's preference for is described as the game time increases.

The game situation $Situation$ refers to the comparison of two forces. There are two levels of comparison, the relationship between your own forces and your own basic decision-making strategies. When the strength of one's own side does not meet the basic needs of resource collection, it is better to send units first to collect resources rather than attack. The second level is the comparison of the strength of the enemy and the enemy. When the strength of the enemy is far greater than the strength of one's own, it should be rather defensive rather than offensive.

Based on the above three aspects, we dynamically adjust the calculation of weights $f\_e(p,t,evaluation(s))$ as follow:

$$f\_e(p,t,evaluation(s)) = g(G(\max,\min,p)\cup\frac{t}{\max\_time})\cdot evaluation(s) \quad (2)$$

$g(x)$ stands for the union relation between $G(\max,\min,p)$ and time proportion, $evaluation(s)$ stands for the contrast evaluation in state s.

The process of dynamic weight generating algorithm is as follow: Line 1-3 show that, player obtain the game time t and player principle p under the evaluated state. Line 4-10 show that player get the evaluation of its own evaluation(s, max) and its opponent's evaluation(s, min). Line 11-13 show the generation of evaluating weights under p, t and evaluation(s).

Algorithm 1 Dynamic Weight Generation(s)

```
1.   While state s need to be evaluated
2.       Get the game time denoted as t
3.       Get player principle denoted as p
4.       For unit in unit(s)
5.           If unit belongs to max then
6.               add unit to evaluation(s, max)
7.           Else If unit belongs to min then
8.               add unit to evaluation(s, min)
9.           End If
10.      End For
11.      dynamic_weight= f_e(p, t, evaluation(s))
12.  End While
13.  Return dynamic_weight
```

4 Experiments and Result

In order to verify the proposed algorithm, an empirical study based on µRTS game is carried out, and the performance of DHEN is compared to the performances of other state-of-the-art evaluating algorithms developed for RTS games.

4.1. Experiment Environment and Setting

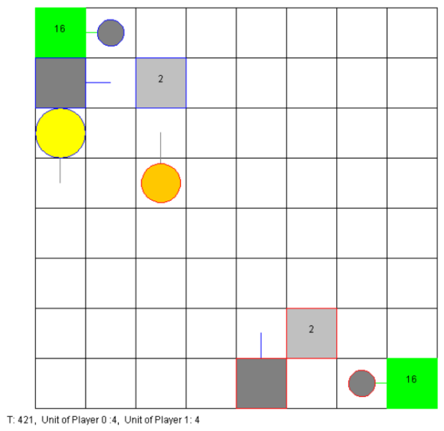

Fig 2. A screenshot of the µRTS game environment. The two players are distinguished according to the blue and red outline colors. The green squares are resources. The white squares are bases. The gray circles are workers, the yellow circles are heavy attackers, and the orange circles are light attackers.

We evaluated the performance of DHEN algorithm using the free-software µRTS (https://githubs.com/ santiontanon/ microrts), which has been used by several researchers to validate new algorithms for RTS games. Figure 2 shows a screenshot of a µRTS game, in which two players compete to destroy opponent's units.

The maps used in our experiments are three: M1 (8 × 8 tiles), M2 (12 × 12 tiles), and M3 (16 × 16 tiles). Maximum game time of M1 is limited to 3000 cycles, of M2 is 3000 cycles, and of M3 it is to 10000 cycles.

To evaluate each algorithm, we conduct a round-robin tournament, in which each algorithm plays 50 games against all other algorithms in each of 3 different maps. The method used to compute the score of each algorithm is: the winner of each game is awarded 1 point, and both algorithms are awarded 0.5 points in the event of a tie. Each of the two AI players in all competitions begins with a single base, an equivalent resource value, and a single worker.

4.2. Experimental Results and Analysis

In this section, we compare the performance of DHEN with other evaluating algorithms in terms of the average score in three maps. We compared DHEN algorithm with simple LTD evaluating function, optimal LTD evaluating function and lanchester evaluating function by applying them in AHTN planning algorithm and IDABCD planning algorithm.

Figure 3 and Figure 4 presents the comparison of the average scores obtained for using different evaluating algorithm in AHTN and IDABCD algorithm. According to the results, applied in both planning algorithm, the DHEN evaluating function can achieve better results. Since the AHTN algorithm has more approaches and depending heavier on evaluation function, the preference is better. In addition, the performance of DHEN varies little with the increasing scale of the maps. The performances of both players deteriorate in map M1 because the map is relatively smaller, so the difference between optimal action and normal action is not obvious for results. With respect to other algorithms, we note that Lanchester evaluation function and optimal evaluation function do not achieve an anticipant performance as it performed in MCTS algorithm. But both algorithms perform better in AHTN algorithm, reveals that the optimal parameters is changing when applied in different planning algorithms.

Table 3. Average Scores of each HTN algorithm with different domain knowledge.

Planning Algorithm	Simple eval.	Optimized eval.	Lanche ster	DHE N
IDABCD	81.6	80.3	73.3	84.3
AHTN	78.3	73	79.3	89

Table 3 shows the average scores of each evaluating algorithm applied in IDABCD and AHTN. We can find that the DHEN algorithm applied in IDABCD improves win rate about 3% compared with simple evaluation function, which performs best among the three benchmark algorithms. The performance of DHEN applied in AHTN brings an improvement of winning rate about 12% compared with the lanchester evaluating function. We can find that, compared with planning process, the evaluation time cost little time.

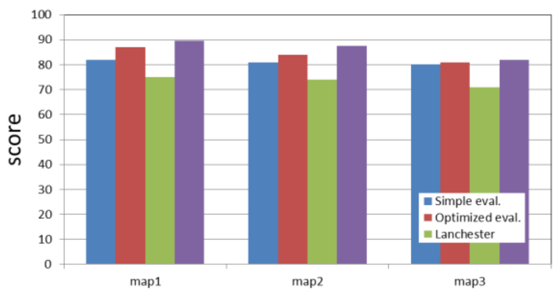

Fig 3. The average score of each evaluation used in AHTN algorithm for map M1, M2, M3 with respect to the CPU time 100 ms.

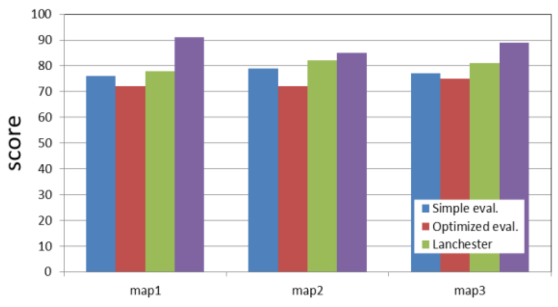

Fig 4. The average score of each evaluation used in IDABCD algorithm for map M1, M2, M3 with respect to the CPU time 100 ms.

Figure 5 shows the average decision times of four evaluating functions. Since the AHTN algorithm is flexible than IDABCD, it needs more decision time. We find that DHEN cost almost the same time as simple LTD evaluating function and optimal LTD evaluating function, because they are calculated according to the same evaluating structure. Since lanchester evaluating function is an iterative process, it needs more time. The DHEN algorithm increase the decision time about 10% compared with simple/optimal LTD algorithm. Therefore, the modification in DHEN brings little time increment compared to LTD evaluation algorithm, which is within an acceptable range.

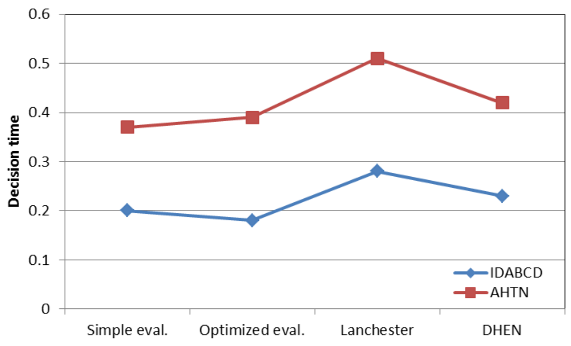

Fig 5. The average decision times of DHEN applied in IDABCD and AHTN.

5 Conclusions

In summary, we have performed both experimental and theoretical study of evaluating RTS game state. A dynamic hierarchical evaluation network is proposed to handle the evaluation problem accompany with the planning process. The experimental results in µRTS game successfully verify the algorithm's validation.

In future work, the DHEN algorithm can be extended in multiple directions. Our experiments demonstrate that domain knowledge of the HEN has a significant influence on the performance of DHEN. However, encoding perfect evaluation network is a difficult and time-consuming process. The automatic extraction of HTN domain knowledge from thousands of RTS game replays may provide an efficient approach.

In addition, we note that using deep learning algorithm to calculate the weights of factors may improve the veracity of planning. Therefore, a modified DHEN algorithm using learning method may enhance its performance in huger game environment.

Acknowledgment

The work described in this paper is sponsored by the National Natural Science Foundation of China under Grant No. 61473300.

References

1. Buro, M. Call for AI research in RTS games. In Proceedings of the AAAI-04 Workshop on Challenges in Game AI (2004), pp.139-142.
2. Ontañón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., & Preuss, M. (2013). A survey of real-time strategy game ai research and competition

in starcraft. IEEE Transactions on Computational Intelligence & Ai in Games, 5(4), pp. 293-311.

3. Ontañón, S. Experiments with game tree search in real-time strategy games. *Artif. Intell.* 2012, arXiv:1208.1940.
4. Chung, M., Buro, M., & Schaeffer, J. (2005). Monte Carlo Planning in RTS Games. IEEE Symposium on Computational Intelligence and Games (pp.117-124). DBLP.
5. Balla, R.K.; Fern, A. UCT for tactical assault planning in real-time strategy games. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*; AAAI Press Palo Alto,, CA, USA, 2009; pp. 40–45.
6. Churchill, D.; Saffidine, A.; Buro, M. Fast Heuristic Search for RTS Game Combat Scenarios. In Proceedings *of the Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*; AAAI Press: Palo Alto, CA, USA, 2012, pp. 112-117
7. Ontañón, S. The combinatorial multi-armed bandit problem and its application to real-time strategy games. In *Proceedings of 9th Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*; AAAI Press: Palo Alto, CA, USA, 2013.
8. Shleyfman A, Komenda A, Domshlak C. On Combinatorial Actions and CMABs with Linear Side Information[J]. Frontiers in Artificial Intelligence & Applications, 2014, 263:825-830.
9. Buro, M. (2015). Adversarial hierarchical-task network planning for complex real-time games. International Conference on Artificial Intelligence (Vol.8, pp.1652-1658). AAAI Press.
10. Kovarsky, A., & Buro, M. (2005). Heuristic search applied to abstract combat games. Lecture Notes in Computer Science, 3501, pp. 66-78.
11. TungDucNguyen, KienQuangNguyen, & Ruck, T. (2015). Heuristic search exploiting non-additive and unit properties for rts-game unit micromanagement. Journal of Information Processing, 23(1), pp. 2-8.
12. Stanescu, M., Hernandez, S. P., Erickson, G., Greiner, R., & Buro, M. (2013). Predicting army combat outcomes in StarCraft. AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (pp.86-92). AAAI Press.
13. Erickson, G., & Buro, M. (2014). Global state evaluation in StarCraft. Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (pp.112-118). AAAI Press..
14. Uriarte, A., & Ontañón, S. (2014). Game-tree search over high-level game states in RTS games. AIIDE. pp.73-79.
15. Bakkes, S., Spronck, P., & Herik, J. V. D. (2013). Phase-dependent evaluation in rts games.
16. Li, Y. J., Ng, P. H. F., & Shiu, S. C. K. (2015). A fast evaluation method for rts game strategy using fuzzy extreme learning machine. Natural Computing(3), pp. 1-13.
17. Stanescu, M., Barriga, N. A., Hess, A., & Buro, M. (2017). Evaluating real-time strategy game states using convolutional neural networks. Computational Intelligence and Games. IEEE.
18. Malik Ghallab, Dana Nau, Paolo Traverso. Automated Planning: Theory and Practice. 2008. Elsevier(Singapore) Pte.Ltd.
19. Sacerdoti, E. D. (1975). The nonlinear nature of plans. International Joint Conference on Artificial Intelligence (pp.206-214). Morgan Kaufmann Publishers Inc.
20. David Churchill. [D]. Heuristic Search Techniques for Real-Time Strategy Games. University of Alberta. 2016.