Minimax Guided Reinforcement Learning for Turn-based Strategy Games

Conference Paper · May 2014

DOI: 10.1109/ICoICT.2014.6914068

2 authors:

Sulaeman Santoso

Universitas Kristen Maranatha

8 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Iping Supriana

Bandung Institute of Technology

265 PUBLICATIONS **222** CITATIONS

SEE PROFILE

Minimax Guided Reinforcement Learning for Turn-based Strategy Games

Sulaeman Santoso
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
sulaeman.santoso@gmail.com

Iping Supriana
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
Iping@informatika.org

Abstract—**Games are a good medium for artificial intelligence (AI) research, since they compare user and machine behavior directly. AI in games is required to imitate human behavior. Though nowadays the use of rule base scripting and hard code behavior is still dominant in commercial game, the use of learning algorithm can be an alternative because of its ability to adapt to changes and to achieve substantial result at a moderate time. This paper investigates the use of learning which is derived from dynamic scripting to provide action in a turn-based strategy game. The algorithm is then combined with the Minimax algorithm to achieve a better performance. The performance of the proposed algorithm is evaluated through a series of matches against a static manually designed AI. The result shows that the proposed algorithm is able to adapt the static AI at a shallow Minimax depth. The algorithm also shows the ability reducing calculation time and using less memory space. It is concluded that Minimax guided reinforcement learning can be applied to the turn based strategy genre.**

Keywords— Artificial intelligence, reinforcement learning, Minimax algorithm, turn-based strategy

I. Introduction

Although the core AI tasks in games is to find a better solution of a problem, developing an AI for games requires that a viable solution should be found at limited time and space. It is common in the game industry to compensate this problem with hard coded behavior or rule based scripting to simulate intelligent behavior in games. These approaches however, have their weaknesses. A static script or behavior may contain flaw that can be exploited by the player [1]. This paper proposed an alternative algorithm which can deal with such problem.

Different types of games required different types of AI. Several genres like the action games require little to no AI whatsoever, but other genres like the strategy genres require an intelligent AI. One of the challenging types of games for artificial intelligence in this genre is the turn based strategy.

In a turn-based strategy games (TBS) players take turn in playing. The TBS actually provides the AI a longer "thinking time" [2] which in turn is both beneficial as the algorithm can calculate more, and at the same time, the user expect a more intelligent AI. Although most TBS have 4 major features (Exploration, Expansion, Exploitation, and Extermination) [2], the focus of this paper is only one aspect of the game, which is extermination. Several works already conducted on AI in TBS uses online learning algorithm, more specifically the reinforcement learning. Reinforcement learning is an algorithm where punishment and rewards are given to shape the algorithm to provide good result. [3].

The outline of this paper is as follow. Section 2 discusses related works being done in AI for TBS. Section 3.1 will describe the game description and environment. Section 3.2 will describe the proposed algorithm. Section 3.3 will describes the testing and experiment being done. Section 4 will describe the conclusion and future works...

II. RELATED WORKS

There are several works implementing AI in turn based strategy which included the use of Case Based Reasoning in 2005 Sanchez-Pelegrin, Gomez Martin, Thomas R. Hinrichs and Kenneth D. Forbus using analogical learning for optimizing food production in turn based strategy games [4]. and Diaz-Agudo implement CBR for a CIVILIZATION clone [5].

One of the most recent works in reinforcement learning for TBS games are in 2009 when Stefan Wender evaluate the uses of Q-learning, Q(γ), one-step sarsa and sarsa(γ) reinforcement learning. This work has shown that reinforcement learning performs better than the original deterministic AI of CIVILIZATION (a turn based strategy game) in city site selection [6].

Before that, Maurice Bergsma and Pieter Spronck in 2008 created the ADAPTA architecture based on task decomposition and machine learning to outperform static opponent and managed to do so [2].

Before ADAPTA Pieter Spronck and colleague in 2005 designed an online learning algorithm called dynamic scripting that took advantage of combining several pre-built line of script in order to adapt to the enemy [7].

Dynamic scripting is one of the interesting reinforcement learning algorithm developed, There are many improvement being made to dynamic scripting algorithm, in 2004 an enhancement to clear outliers are developed [8] and a difficulty scaling feature is applied [9], in 2007 automatic rule ordering for dynamic scripting is introduced [10].

The proposed algorithm is basically based on the automatic rule ordering of dynamic scripting, but it combined static rule list with dynamic rule ordering through reinforcement learning. In later stages of the study the algorithm becomes a combination of reinforcement learning with the Minimax algorithm. The Minimax algorithm is an adversarial search algorithm commonly used [11].The proposed algorithm is then applied to a turn based strategy game against a static manually designed AI.

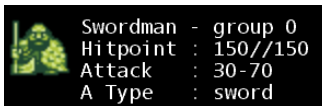
III. Methodology

Before a proposed algorithm could be describe, a game environment and rules are defined in order to sufficiently test the algorithm.

A. *Game Environment and Rules*

The environment to test the proposed algorithm is a simplified turn based strategy games based on commercial turn based strategy game with the following specification:

1. Game board are 23x23 tile with no obstacle
2. The objective of the game is elimination of the other party (which is attained when all member of the enemy party has less or 0 health point)
3. There are four types of unit in the game: Swordman, spearman, axeman, and arbalest. Each of this four unit has different stats



```
Swordman  - group 0
Hitpoint  : 150//150
Attack    : 30-70
A Type    : sword
```

Fig 1 Character Stat

4. Every melee unit has weakness and strength over the other type of unit. Swords beats axe, axe beats spear, and spear beats sword. If a swordman attacks an axeman, the attack will not likely to miss (half miss chance). But if a swordman attack a spearman the attack will be most likely to miss (double miss chance).
5. Arbalest is the only range unit in the game and is able to attack through obstacle or other unit
6. A team of player consist of 2 unit of every type.
7. Unit cannot move through other unit (ally of foe alike)

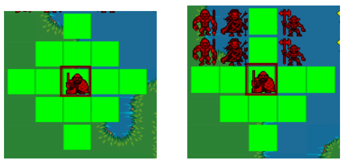


Fig 2 Unit movement

The test is conducted by the proposed algorithm with a static AI in 300 matches with similar unit composition and static unit arrangement. The average winning rate of every 100 matches and the average calculation time of the Minimax algorithm will be noted to evaluate the quality of the proposed algorithm.

The static AI used in the testing environment is a "rush" algorithm. This behavior can sometimes be an effective algorithm in several cases in turn based strategy [7]. Fig 3 shows the behavior of the static algorithm.

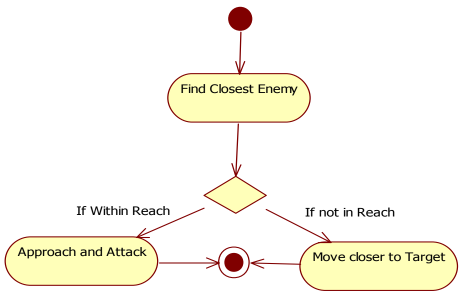


Fig 3 Static AI Logic

B. *Proposed Algorithm*

The proposed algorithm originated from an attempt to alter Spronck's dynamic scripting algorithm. Spronck's dynamic scripting chooses the line of script to be used in the game, [7] the modification was to choose not the line of script but the action carried out in each turn. Though in dynamic scripting algorithm every unit has a different rule base, the proposed algorithm only split the action list into two. The action lists are melee unit rule and ranged unit rule. This is because the game environment in this paper is a lot simpler than those of dynamic scripting test done by Spronck. In every action list there are a total of 11 actions to choose, based on domain knowledge. The 11 action rules are:

1. Attack closest
2. Attack farthest
3. Attack weakest
4. Attack opposites
5. Attack range
6. Maneuver to safe area
7. Scatter
8. Flock with teammates
9. Maneuver to aggressive position
10. Maneuver to attack mutual target
11. Maneuver to position just outside of enemy's reach.

Each one of these rules has a weight value that states how likely the rules are to be chosen in a turn. The modified reinforcement learning seeks to find the best combination of weight to create an intelligent behavior to battle the static AI. The reinforcement learning algorithm is depicted in Fig 4.

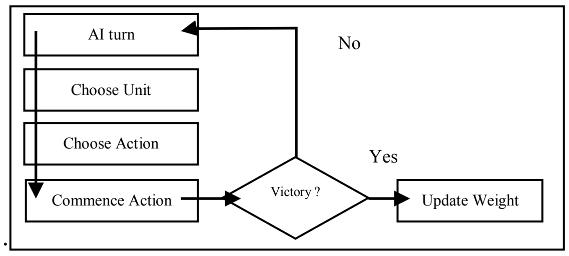


Fig 4 Static AI flowchart

At each turn the algorithm will provide each unit with an action that is taken from the action list randomly with a consideration of the action's weight. Action with a less weight will less likely to be chosen for the unit action. At the end of a battle, a weight adjustment procedure is activated.

The weight adjustment is adjusted according to how well the team performs in the last match. The team's performance is calculated using the following formula (adapted from dynamic scripting fitness function in [1]):

$$\sum_{i=0}^{g}\left(\frac{H_i}{H_{max-i}}\right) * 1 + \sum_{i=0}^{k}\frac{E_{max-i}-E_i}{E_{max-i}} * 3 \qquad (1)$$

Where H_i is the remaining health point of ally unit i . H_{max-i} is the maximum health point of ally unit i, E_{max} is the total maximum health point, E_i is the remaining health point of the enemy i, k is the number of enemy and g is the number of AI.. The weight's adjustment is then applied to the dominant rule in the match. The dominant rule is decided by calculating the average rule used in the match. Rules that are above average in its uses will be granted a positive reward to its weight while rules that are below average will be punish instead. The result of this learning algorithm after the last 100 match of the 500 match is a winning rate of 70%.

Although the winning rate of the algorithm is quite high and increasing, the fact that it took 500 matches for the rules to be effective is not acceptable. This is due to the non-deterministic factor in deciding action and non-deterministic factor in the game itself. The amount of punishment and rewards also greatly affected the process of learning. A higher number will grant faster convergence at the cost of quality, while a low number will grant slower convergence but a better quality. Fig 5 shows the result of the last 100 matches between reinforcement learning algorithm and static AI. As seen in Fig. 5, the trend lines are increasing but very slowly.

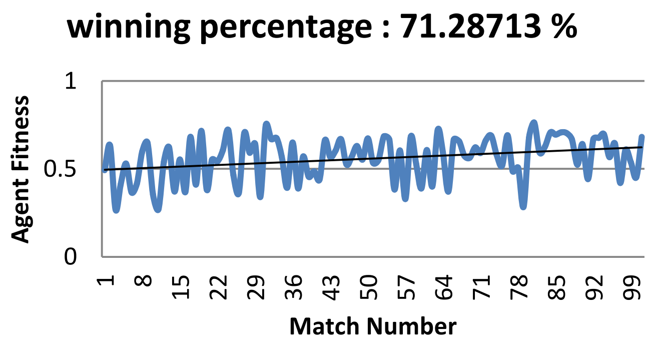


Fig 5 Reinforcement learning result

To prevent early convergence it is proposed to combine the predictive nature of the Minimax algorithm to the reinforcement learning algorithm. Minimax algorithm calculates every player and enemy movement up to a point; this usually means a high number of branching factors. At the testing environment each unit has an average of 20 set of movement which is translated to 20 branching factor. Without optimization, the Minimax algorithm running on a branching factor equal to 20 times unit number is intractable by nature. So it is decided to do an abstraction of the problem, instead of calculating each unit's movement it is decided to focus on selecting group strategies. There are 3 major strategies to pick from the player side, attack strategies, defensive strategies, and maneuver strategies. On the enemy side of the Minimax tree the 11 actions that are used in the reinforcement algorithm to simulate enemy behavior. Every time a player chooses a strategy, each unit will then be assigned an action in the same manner as the modified reinforcement learning algorithm. It is expected that by doing so, rules that are beneficial to the player can be identified in less time. The evaluation function of the proposed Minimax algorithm is a modification of the modified reinforcement learning. Although the Minimax algorithm implemented in this paper includes the alpha beta pruning and depth limit, it does not however implement any other method of optimization.

C. *Test result and Analysis*

In order to prove the quality of the proposed algorithm an evaluation is conducted in a 5 x 100 times game scenarios with different Minimax depth with and without reinforcement learning. The test result shown in Fig. 6 is the average of the 500 matches.

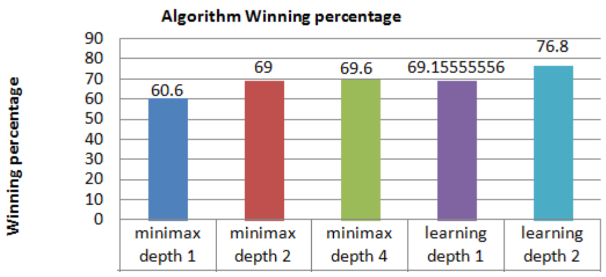


Fig 6 Test result

In Fig. 6 is shown that the learning with Minimax algorithm achieved a 76,8% result at depth 2 which is the same amount of winning percentage as Minimax without learning with depth 4, while Minimax with learning at level 1 surpass the Minimax without learning at level 1. It is also shown from figure 6 that increasing depth of Minimax algorithm only yields menial improvement, while implementing learning algorithm yields a more significant result. This result is substantially better than using reinforcement learning without Minimax since the result in the first 100 match is already stable and in accordance with the next 400 match data as opposed to using reinforcement learning alone; it needed 500 matches to yield 77% of winning rate.

Since the average thinking time of the reinforcement algorithm is not as substanstially different from the Minimax algorithm, the algorithm time are recorded only for the Minimax algorithm without learning.

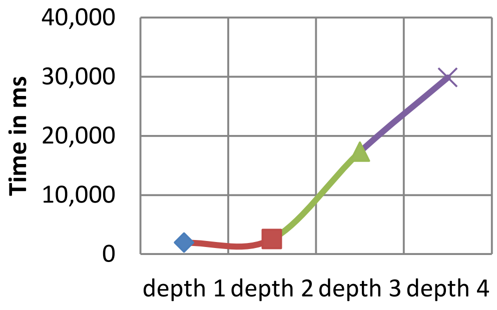


Fig 7. Test result in time (ms)

IV. CONCLUSION

After the test match is conducted with the proposed algorithm pitted against a manually design static AI, It is shown that the proposed algorithm is able to defeat the static AI by a percentage of 76.8%. The proposed algorithm also achieved a better result than the modified Minimax algorithm, the proposed algorithm at depth 2 has more winning rate than the Minimax algorithm at depth 4 which obviously invested more calculation time. This shows that reinforcement learning guided with Minimax algorithm can be an alternative to be use in a turn based strategy game, and that the abstraction of unit movement to strategy is a viable option to reduce branching factor.

Although it is proven that Minimax guided reinforcement learning is capable of adapting to static AI, it is yet to be seen whether or not the AI can stand on itself against a human player. The current works only accommodates the extermination factor of the TBS game. For future work, it would be convenient to include all the 4 factors in TBS and also a dynamic way of determining reward and punishment.

REFERENCES

[1] Nareyek, "Intelligent agents for computer games," in *Computer and games, second international conference*.: Springer-verlag, 2002, pp. 414-422.

[2] Maurice Bergsma and Spronck Pieter, "Adaptive Spatial Reasoning for Turn-based Strategy Games," in *Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, Netherland, 2008.

[3] Tom M Mitchell, *Machine Learning*.: McGraw Hill, 1997.

[4] Thomas R Hinrichs and Kenneth D Forbus, "Analogical learning in a turn based Strategy Game," in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.

[5] Sanchez-Pelegrin , Gomez-Martin , and Diaz-Agudo , "A CBR module for a strategy videogame.," in *1st Workshop on Computer Gaming and Simulation Environments, 6th Internation Conference on Cased Based Reasoning*, 2005.

[6] Stefan Wender, "Integrating Reinforcement Learning," Auckland, 2009.

[7] Pieter Spronck, mark Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma, *Adaptive Game AI with Dynamic Scripting*. Maastricht , 2006.

[8] Pieter Spronck, Ida Sprinkhuizen-Kuyper, and Eric Postma, "Enhancing the Performance of Dynamic Scripting in ," Netherland, 2004.

[9] Pieter Spronck, Ida Sprinkhuizen-Kuyper, and Eric Postma, "DIFFICULTY SCALING OF GAME AI ," Netherland, 2004.

[10] Timor Timuri, Pieter Spronck, and Jaap van den Herik, "Automatic Rule Ordering for Dynamic Scripting," Netherland, 2007.

[11] Peter Norvig and Stuart J Russel, *Artificial Intelligence : A modern approach, 3rd Edition*.: Prentice Hall, 2010.