JIRAE, Vol. 1, No. 1, September 2016, 33-37
e-ISSN 2407-7259
DOI: 10.9744/JIRAE.1.1.33-37

Optimization of Units Movement in Turn-Based Strategy Game

Kristo Radion Purba[a], Liliana[b], Johan Pranata[c]

Department of Informatics, Petra Christian University, Surabaya, Indonesia
[a]kristo@petra.ac.id, [b]lilian@petra.ac.id, [c]tomochin.johan48@gmail.com

Abstract. Each game has an artificial intelligence that is used to fight the player, which will provide more challenge. But in some strategy games, unit movements are usually done using simple considerations. For example the rest of unit lives, unit strength, and so forth. In this study, a turn based strategy game is designed using genetic algorithm to control the movement of the enemy armies. In each turn, the enemy will move based on the potential level of produced damage to and from the opponent, the distance between the units, and the distance to the opponent's building. The genetic algorithm's chromosome for each unit contains the following information: the position where the unit will move, who is the target, and the distance to the armies' centroid. Distance to centroid (midpoint) is used to force the units to remain in the set. The genetic algorithm process is used to control when and where the units will move or attack. From the test results, the genetic algorithm can create a more powerful enemy than the randomly moving enemy because it creates a higher winning chance of enemy units and acts more efficiently, in terms of the usage of money, the damage produced to the opponent, and the received damage.

Keywords: Artificial Intelligence, Genetic Algorithm, Turn-based strategy game, Units Movement.

1. Introduction

Computer game has many types/genre that is currently evolving, such as strategy games. Strategy game is usually played by a human player, organizing buildings and armies, defending its base, and attacking enemy's base. Strategy game is a genre that uses careful planning to achieve victory [1]. Turn-based strategy game, which is the expansion of the strategy genre, grows quite rapidly, because this game genre is challenging; every bad or good move can affect the entire war. The intensity of the graphics, gameplay, mechanical factors, and enemy intelligence are the key to success for this kind of game [2].

Each game has an artificial intelligence implemented in the enemies to fight the player, but in some strategy games, units movements are made based on brief thoughts. For example, the units' remaining lives, the strength of the armies, the enemy units which will be targeted, and so forth, are usually the factors that a human player considers when playing. Strategy game has so many possibilities that can be optimized in terms of enemy AI [3].

This study designed a turn based strategy game that uses a genetic algorithm to regulate the movement of the enemy, with consideration of the condition of all the player's armies and the enemy at a time. The conditions are the potential level of damage to the opponent, the potential damage from the opponent, the distance between the units, and the distance to the opponent's main building.

1.1 Genetic Algorithm

Genetic algorithms are suitable for optimizing a searching space that is uncertain and broad [4]. Genetic algorithm is a stochastic search algorithm based on biological evolution. The basic principle of Genetic Algorithm is the selection, crossover, and mutation of individuals with the aim of generating a better generation than ever before [5]. Here are the steps of the genetic algorithm:

1. Randomization of initial individual.
2. Calculation of the fitness of each individual.
3. Select a pair of chromosome for mating. The selection is done using a roulette method.
4. Do crossover. Every two (a pair of) individuals will be mated with certain chance.
5. Do mutations in individuals with certain chance, and select random genes to be mutated.
6. Do elitism process. The individual with the worst fitness will be replaced by an individual with the best fitness (i.e. the elite) before roulette. This elite is used to adapt with the dynamic environment after the mating processes [6].
7. Current result of individuals will be used for the next iteration as a new population.
8. Steps 2-7 will be repeated continuously, until condition 9 is satisfied.
9. The process of genetic algorithm is terminated after certain conditions, such as the maximum fitness is stable three times in a row.

2. Research Methods

The game designed in this study takes the type of turn-based strategy in which two players take turns controlling their units alternately. The player who succeeds in achieving his/her goal becomes the winner. The game is using grid system (50x50 grids), which means a unit can move or attack based on the grid positions.

The game begins with each player having one main building as a place to make units. The main building also needs to be protected because if the main building belonging to a player is destroyed, the game will end and the player is declared the loser.

If a player is currently in turn, he or she can create units or move units. Created units will appear around the main building. In this game, to limit the searching space, a unit can be created instantly (i.e. no training time) if there is a sufficient amount of money.

The player and the enemy get the same amount of money at the beginning, and on each turn, players earn extra money. The money is used to make the units in the main building. If a unit has killed an enemy unit, the unit's owner receives the money. Players can end a turn by pressing the end turn on the UI (User interface). Once a player ends the turn, the turn will be switched to the opposing player.

2.1 Units Type

In this game, there are various types of units, which have statistics i.e. attack, attack type, attack range and the armor type. In addition, each unit has different creating/training costs. Also, each unit has different repercussions when it attacks certain unit types. There are 3 types of armies based on armor type, i.e. infantry, fortified, and mechanical. There are 3 types of attack, i.e. melee, bullet, and explosive. Relevance of attack type with defense type is as follow:

- Melee attacks will inflict more damage to mechanical armor (+50%). Melee will cause lower damage to fortified armor (-50%).
- Bullet attacks will inflict more damage to infantry armor (+50%). Bullet will cause lower damage to mechanical armor (-50%).
- Explosive attacks will inflict more damage to fortified armor (+50%). Explosive will cause minimal damage to infantry armor (-50%).

Names and statistics of the designed armies can be seen in Table 1.

Table 1. Units Type

Name	Atk	Type	Rng	Armor	HP	Move
Infantry	15	Melee	1	Infantry	20	5
Cavalry	20	Melee	1	Infantry	20	4
Musketeer	25	Bullet	4	Fortified	25	3
Artillery	30	Explosive	4	Mechanical	30	2

Notes:

- Atk = Attack, means the attack damage that is caused to an opponent in a single attack
- Type = Attack type. Melee means close-ranged unit, bullet means ranged unit that attacks the enemy using a hand gun, explosive means ranged unit that attacks the enemy using a cannon or an explosion.
- Rng = Attack range. For example, value 4 means the attack range is 4 grids.
- Armor = Armor type. Infantry means simple armor attached to a human body; fortified means concrete/ building armor; mechanical means the artillery/tank/ other vehicle type armor.
- HP = Health points, means the amount of health. If the HP reaches 0 or below, the unit will be dead.
- Move = Move count. For example, value 5 means it can move up to 5 grids in a single turn.

2.2 Genetic Algorithm for Movement

In the planning of unit movements (attack included), genetic algorithms are used to determine the steps that will be controlled by AI. The desired output is that each unit causes an optimal damage to an opponent unit, and the potential damage inflicted by the opponent should be as low as possible. Optimal damage means the most advantageous (see 2.1) and the shortest move to attack. There are four criteria that must be met:

- Potential damage produced to the opponent units.
- Potential damage received from the opponent units.
- The average distance to the centroid of the entire units.
- The average distance of each unit to the opponent's main building.

The reason in choosing these criteria is to produce maximum damage to the opponent units and to achieve victory faster. Besides the damage produced, the potential damage from the enemy should be a consideration, so that ally units can live longer. The units are required to be in group, not separated too far with each other, because the closer the unit with others, the more quickly the unit can help when the ally units are being attacked by opponent units.

With the criteria that must be fulfilled in the AI, the resulting chromosome for genetic algorithm design is shown in Figure 1.

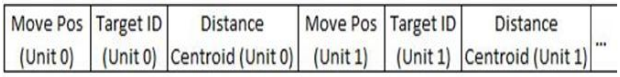

Figure 1. Individual design for units

In the design of the individual, the length of the chromosome is adapted to the number of units owned by AI. There are three parts of the chromosome, which are the position of the movement to be performed (Move Pos), the ID of the enemy unit to be attacked (Target ID), and the unit distance to the centroid of entire units (Distance to Centroid). The example of individual design can be seen in Figure 2.

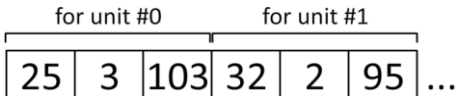

Figure 2. Example of individual design for units

Notes:

- The first 3 genes represent the unit index 0, next 3 genes represent unit index 1, and so on.
- The number 25 (1st gene) means, the unit #0 will move to grid ID 25. Each grid in the game is given an ID. The top left grid is given ID 0, the right of it is 1, continuing horizontally to the right side, before continuing to the next row.
- The number 3 (2nd gene) means, the unit #0 will attack opponent unit #3. Each opponent unit is given an ID.
- The number 103 (3rd gene), means the unit position after moved to grid #25 (1st gene) will have the distance of 103 grids to the centroid.

The following steps of implementation of unit movements using genetic algorithms can be explained in sub section 2.2.1 up to 2.2.9.

2.2.1 Generate Individuals

In the first generation, each individual will have randomized genes. There will be 10 individuals generated. The results of random numbers will be converted into the position of the movement to be performed, while the distance from the centroid is calculated using Euclidian distance.

In determining the Target ID, all opponent units will be checked if any of them is in the unit's attack range. If any opponent unit is in range, the AI will prioritize the target that can be killed with one hit. When an enemy unit is killed, AI earns money.

2.2.2 Fitness Value Factors

The potential damage (to and from opponent units), distance to enemy building, and distance to centroid are values that contribute to the fitness value. The average distance of each unit to the opponent's main building can be calculated using the mean of Euclidean distance, shown in the following formula:

$$dist = \frac{\sum_{i=1}^{n}\sqrt{(x_b - x_i)^2 + (y_b - y_i)^2}}{n} \quad (1)$$

Notes:
- n = Units count
- x_i and y_i = The (x,y) position of unit-i
- x_b and y_b = The (x,y) position of opponent's main building

The potential level of damages from the opponent units is calculated from averaging the level of damage of all opponent units against each ally unit in the chromosome, also considering:
- Number of turns required to attack the enemy unit. Fewer turns are better.
- Attack range of unit. Higher range is better, because it has a higher chance to attack.

The formula for potential damage from opponent units is shown on the following pseudo code.

```
selfLoss = 0
Loop opponent units (as i)
     Loop ally units (as j)
          selfLoss += dmg(i,j) × (1/turn(i,j)) ×
rng(i)
     Next j
Next i
selfLoss /= n
```

Notes:
- selfLoss = Potential damage from opponent units
- dmg(i,j) = The amount of damage that is caused by enemy-i to enemy-j, considering the attack type and armor type (see chapter 2.1)
- turn(i,j) = Number of turns that is required for enemy-i to attack enemy-j (move to reach the target and then attack).
- rng(i) = Attack range of enemy-i.

The potential damage to opponent units (enemy loss) is using the same formula, but the unit side is reversed.

The third factor that contributes to the fitness value is the average distance of units' position to centroid, and it is calculated using the following formula:

$$avgToC = \frac{\sum_{i=1}^{n}\sqrt{(x_m - x_i)^2 + (y_m - y_i)^2}}{n} \quad (2)$$

Notes:
- n = Units count
- x_i and y_i = The (x,y) position of unit-i
- x_m and y_m = The (x,y) position of centroid, calculated by averaging X of all ally units and averaging Y of all ally units

2.2.3 Fitness Function

Calculation of fitness in each individual is based on the following formula:

$$f = \frac{1 + 10 \times enemyLoss}{5 \times dist + 0.2 \times selfLoss + 2 \times avgToC} \quad (3)$$

The formula can be described as follows:
- *enemyLoss* represents the total potential damage produced to the enemy. This variable is multiplied by 10, which means it is valued the most.
- *dist* represents the average distance of each unit in the individual to the opponent's main building. *dist* becomes a factor that lowers the individual fitness, because the greater the distance, the more difficult it is for the AI units to siege the opponent's main building.
- *selfLoss* represents the total average of potential damage that all opponent units inflict on the units of the current individual. *selfLoss* is a factor that lowers the individual fitness, because the higher the number of attacks received by AI units, the greater the danger that awaits.
- *avgToC* represents the average distance of each AI in an individual against a centroid (midpoint). The farther the average distance, the more difficult it is for the AI units to perform backups on the besieged units.

The higher the fitness value generated, the better the individual is.

2.2.4 Roulette Selection

The selection on the individuals uses the roulette wheel to define the next generation. Only the top 5 (best fitness) of the population (50% of total population) will enter the roulette to speed up finding a solution. Roulette wheel method uses fitness as the probability of a chromosome being selected in the next generation. The following formula determines the chance of a chromosome to be selected in the roulette:

$$Prob_{Indiv} = \frac{Fitness_{Indiv}}{Fitness_{Total}} \quad (4)$$

The roulette will choose 10 chromosomes that will go to the next step. The chromosomes will be chosen based on the probability. For example, the probability of the top five is (35, 25, 20, 10, and 10). These top five will be picked using roulette until the count of chromosome is 10. Thus, the same chromosome can be picked more than once. The roulette can have the following results: 3,1,2,1,4,0,2,4,1,2 (unit index).

2.2.5 Crossover

The GA process will do the crossover between two individuals (a pair) with a 70% chance. The crossover occurs by choosing two random integers (from one to the number of genes) as the start index and the end index. Then, between the two individuals, genes from start to end index will be exchanged, resulting in two new individuals in each copulation.

2.2.6 Mutation

After the crossover, each individual will be mutated with a 10% chance. The mutations are performed on Move Pos variable of each unit on the individual. Since the Move Pos is the index of grid, there is a chance that the chosen grid is an area that cannot be occupied (because of tress/rocks, etc.). In this case, the index of grid will be re-randomized. Once the mutation is done, Target ID, Distance to Centroid, and the final fitness value will be recalculated, due to the possibility that their values change.

2.2.7 Elitism

Elitism is done by adding one chromosome with the highest fitness value to replace the worst chromosome in the next generation. This process occurs by a 100% chance.

2.2.8 Iteration

Step 2.2.2 to 2.2.7 will be repeated until it meets the specified number of generations. The maximum number of generations used for the unit movement are 20 generations, or it will stop if the maximum fitness is stable for three generations consecutively. Twenty is chosen as the number of generations because if it is higher, the process will be too CPU intensive, causing a lot of lag. Based on 10 trials of the GA, eight of them stop at iteration around 15-18, because the fitness value is stable. So, the limit of 20 is adequate for limiting the CPU usage.

2.2.9 Final Result

At the final stage, the best individual (maximum fitness) of the last generation is picked out and used to determine AI actions on any unit movement.

3. Result and Discussion

This section will discuss the testing procedures and the results. Genetic Algorithm AI of the game will be compared with the random AI method, where the opponent moves randomly. Comparisons were made covering several aspects, i.e. comparison of the financial terms, produced and received damage, created and killed units, and number of turns to win.

3.1 Player Battle Test versus AI (AI uses Infantries and Cavalries only)

This chapter is used to test the genetic algorithm of the AI where the player can use all unit types, whilst the AI (enemy) can only use infantries and cavalries. The test is carried out until the game is finished, and any data generated in the game will be logged. The test is done using two types of AI: the random AI and genetic algorithm AI. Game results are recorded in Table 2.

Table 2 compares the results of testing both methods of AI. In the aspect of the unit created, it is evident that genetic algorithm was capable of suppressing the number of units made by the player with a fewer number of units than that required by random AI. In the aspect of unit loss, random AI was not able to defeat any player unit, while genetic algorithm could beat eight of them, with the number of units lost far less than the AI Random.

In the aspect of money used, the player used more money battling against AI genetic algorithm than AI random, but AI genetic algorithm used less money than random AI did. In the aspect of money earned, there is a drastic change. The money generated by the player opposing the AI genetic algorithms was much less than when the player battled against AI random. Similarly, the AI genetic algorithm could earn more money than AI random did.

In the aspect of damage, the damage produced by the player's units was much lower than the damage produced by the enemy's, when the player was battling against the AI genetic algorithm.

In the aspect of the number of turns in this test, there was no difference.

Table 2. Game Result – Player versus Infantries and Cavalries

Aspect	AI Random		AI Genetic Algorithm		Increase Value*	
	Player	Enemy	Player	Enemy	Player	Enemy
Unit Created	13	19	8	13	-5	-6
Unit Dead	0	17	8	3	8	-14
Money Used	265	250	200	245	-65	-5
Money Earned	810	330	319	724	-491	394
Damage Produced	778	156	134	525	-644	369
Damage Received	156	778	525	134	369	-644
Total Turn	14		14			
Winner	Player		Enemy			

* The increase value is obtained by comparing the results of both AI. Comparison is done by finding the difference of AI Random and Genetic Algorithm.

Table 3. Game Result – Player versus All Unit Types

Aspect	AI Random		AI Genetic Algorithm		Increase Value*	
	Player	Enemy	Player	Enemy	Player	Enemy
Unit Created	22 (9.17)	23 (9.58)	7	14	-2.17	4.42
Unit Dead	14 (5.84)	24 (10)	8	2	2.16	-8
Money Used	645 (268.75)	610 (254.17)	210	300	58.75	-45.83
Money Earned	1157 (482.08)	691 (287.92)	177	895	-365.08	607.08
Damage Produced	1054 (439.17)	622 (259.17)	126	550	-313.17	290.83
Damage Received	622 (259.17)	1054 (439.17)	550	126	290.83	-313.17
Total Turn	36 (15)		15			
Winner	Player		Enemy			

3.2 Player Battle Test versus AI (AI uses All Unit Types)

This chapter is used to test genetic algorithm of the AI where the player and the enemy can use all unit types. The human player would attempt to make the units in the same orders as what the player did in the previous test (chapter 3.1). The results can be seen in Table 3. It compares the results of testing of both methods of AI. The numbers of turns for AI random and genetic algorithms were not the same; 36 versus 15. Because of the difference, values for the AI random would be normalized to assume it were using 15 turns. The numbers in parentheses are the normalized values. The normalization is done to make the comparison easier.

In this test case, AI random could last up to 36 turns, but because of regulation on the manufacturing units and unit movements, AI random could not make a decision carefully, so the AI random was defeated. In contrast, the AI genetic algorithm was able to produce results that were much better than the AI random.

4. Conclusion

During the planning, analysis, design, and programming through journal writing, there are several conclusions which can be derived:

- The use of genetic algorithm of AI is more suitable when it is used in the turn-based strategy genre (rather than the real time strategy), because of the time to carry out the process of picking a best move that requires a long time.
- Applying genetic algorithm in a game is influenced greatly by the statistics of every unit in the game, so the AI should be tested thoroughly. Also, due to the very broad searching space of a strategy game, the GA can be tweaked more.
- The use of genetic algorithms for AI is proved to be better than the random method, because the AI can review all of its and the player's conditions.
- This research can be further expanded by adding more clans with more variations of units to make the optimization process even more challenging. Adding air, ground, and sea units are also interesting.
- The GA in this research can be combined with another AI method, such as Fuzzy and Neural network, to make the optimization process faster, because predetermined expert data is included.

References

1. Adams, E., *Fundamentals of Game Design*, 3rd ed., New Riders Publishing, Thousand Oaks (CA, USA), 2013.
2. Fabricatore, C., Gameplay and Game Mechanics Design: A Key to Quality in Video Games, *Proc. of the OECD-CERI Expert Meeting on Videogames and Education*, Santiago (Chile), Oct 2007.
3. Weber, B.G., Mateas, M., and Jhala, A., Building Human-Level AI for Real-Time Strategy Games, *Proc. of the AAAI Fall Symposium on Advances in Cognitive Systems*, Arlington (VA, USA), Nov 2011, pp. 329–336.
4. Sivanandam, S.N. and Deepa, S.N., *Introduction to Genetic Algorithms*, Springer, Berlin Heidelberg (Germany), 2008.
5. Negnevitsky, M., *Artificial Intellegence: A Guide to Intelligent Systems*, 3rd ed., Pearson Education, Canada, 2011.
6. Yang, S., Genetic Algorithms with Elitism-Based Immigrants for Changing Optimization Problems, *Proc. of EvoWorkshops 2007*, Valencia (Spain), LNCS: 4448, Apr 2007, pp. 627–636.