

# Optimization of Auto Equip Function in Role-Playing Game Based on Standard Deviation of Character's Stats Using Genetic Algorithm

Kristo Radion Purba

Informatics Engineering - Petra Christian University  
Surabaya, Indonesia  
Phone: +628123577791  
Email: kristo@petra.ac.id

**Abstract.** Genetic algorithm is a well-known optimization solution for an unknown, complex case that cannot be solved using conventional methods.

In Role-Playing Games (RPG), usually the main features are character's stats and equip items. Character has stats, namely strength, defense, speed, agility, life. Also, equip items that can boost character's stats. These items retrieved randomly when an enemy dead.

A problem arise when the player have so many items that we cannot choose the best. Latest items doesn't always mean best, because usually in RPGs, items don't always boost all stats equally, but often it reduces certain stat while increasing the other.

Based on this, a function is built in this research, to auto equip all items, based on the standard deviation of character's stats after equipping. The genetic algorithm will evaluate the best combination of gloves, armors and shoes. This algorithm involves the process of evaluating initial population (items combination), selection, crossover, mutation, elitism, creating new population. The algorithm stops when the best fitness is getting stable in successive 3 generations.

After the auto equip process, the character is getting significantly stronger compared to using default equip items, measured by the remaining life after fighting with several enemies.

Keyword: Genetic algorithm, RPG game, optimization, equip item, artificial intelligence, standard deviation

## 1 Introduction

Computer games, or better known as game is an application that uses electronic media, is a form of multimedia entertainment that is made interesting. Developing realistic characters and intelligent enemy is the main task in the modern game design, in particular, real-time strategy game (RTS), first-person shooter (FPS), and role-playing games. Non-player characters (NPC) must be able to provide enough challenge to attract players to defeat opponents (Lin, 2011:325), therefore the NPC should be given

an intelligence. Artificial intelligence is the ability of computer to solve a problem that requires ingenuity if done by humans (Johnson, 2001).

The game designed in this research will implement a genetic algorithm (GA) to optimize equipment used by the enemy and player. GA is a type of artificial intelligence. Each enemy in this game has 3 types of equipment, namely gloves (glove), body protection (armor) and footwear (shoes). The GA will optimize equipment used by the enemy, so that the status of enemy forces evenly.

Besides the optimization done for the enemy, there is also auto equip function that works for the player character. The auto equip uses GA to find the best combination of equipment so that the standard deviation of character stats can become as small as possible. The use of standard deviation, is because of having to high in certain stats while very low in other stats often feels not balanced.

The use of auto equip function for player character will make players easier to play, while expert players still able to equip manually. The optimization done for the enemies are also contributing to the game's complexity and variation.

## **2 Literature Review**

This research was conducted with reference to the studies that have been done before. To give some knowledge in general, concepts about role-playing game (RPG) and genetic algorithm will be explained in this chapter.

### **2.1 Role-Playing Game**

A role-playing game (RPG) is a genre of video game where the gamer controls a fictional character (or characters) that undertakes a quest in an imaginary world. (Janssen, n.d.). Defining RPGs is very challenging due to the range of hybrid genres that have RPG elements. Traditional role-playing video games shared three basic elements:

- Levels or character statistics that could be improved over the course of the game
- A menu-based combat system
- A central quest that runs throughout the game as a storyline

Modern and hybrid RPGs do not necessarily have all of these elements, but usually feature one or two in combination with elements from another genre.

### **2.2 Genetic Algorithm**

In the computer science field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a meta-heuristic) is routinely used to generate useful solutions to optimization and search problems. (Mitchell, 1996:2) Genetic algorithms belong to the larger class of evolutionary algorithms (EA). The mechanism of the EA is inspired by the evolutionary systems in biology, such as reproduction, mutation, recombination and

selection. EA is divided into several techniques, which is genetic algorithm, genetic programming, evolutionary programming, etc. (Ashlock, 2004:9).

Genetic algorithms are something worth trying when everything else has failed or when we know absolutely nothing of the search space (Sivanandam, 2008: 22). In genetic algorithms, the term chromosome typically refers to a candidate solution to a problem, often encoded as a bit string. The "genes" are either single bits or short blocks of adjacent bits that encode a particular element of the candidate solution.

The general structure of a GA in computer science can be defined by the steps as follows (Basuki, 2003: 3):

- Generating the initial population, the initial population is generated randomly to obtain the initial solution.
- Selection of the individual, this process will evaluate each population to calculate the fitness value of each chromosome and evaluate it until stop-ping criterion is met. If the stop criterion is not met then the new generation will be formed again by repeating step 2.
- Reproduction and form a new generation, in the form used three operators mentioned above, namely reproduction operator/selection, crossover and mutation. This process is performed repeatedly to obtain a sufficient number of chromosomes to form a new generation in which this new generation represents the new solution.
- Crossover, used to cross the genes between couple, using certain probability. Crossover probability is a process to indicate a ratio of how many couples will be picked of mating. (Vivek and Narayanan, 2013:233)
- Mutation. In a binary representation, a mutation consists of flipping bits with a particular probability (Vivek and Narayanan, 2013:230)
- The process of elitism  
This process ensures that the generation of a population is not declining quality from generation to generation. This process will replace the worst individuals of the new population with the best individual in the population prior to reproduction.
- Stop at a certain generation.  
Stop after a few consecutive generations obtained the highest fitness value is not changed. Stop when the next generation n is not obtained higher fitness values.

### 2.3 Standard Deviation

The variance of a set of data from the mean that is, how far the observations deviate from the mean. This deviation can be both positive and negative, so we need to square these values to ensure positive and negative values do not simply cancel each other out when we add up all the deviations. (Math Centre, 2003:1). Standard deviation is the root of variance. The standard deviation, will be measured in the same units as the original data. The equation for standard deviation is given at equation (1).

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (1)$$

### 3 Design Overview

In this chapter, will be explained about the game system is designed, and also how the application of the algorithm.

#### 3.1 Character Stats

In the game that is designed in this research, the stats (statistics) of the player's character or enemies' characters that will be used are:

- **Attack / atk**  
Attack stat will reduce enemy's life when the attacker attacks its enemy. The maximum value is 99.
- **Defense / def**  
Is the percentage rate that will reduce the amount of incoming attack damage when enemy attacks. The maximum value is 99.
- **Life / life**  
Is the life value. If character run out of life, it will die. The maximum value is 999.
- **Speed / spd**  
Is the number of pixels / frame that represents the movement speed of the character. The maximum value is 99.
- **Agility / agi**  
A percentage of chance a character can dodge incoming enemy attacks. The maximum value is 99.
- **Ammo**  
The number of ammunition owned by enemy archers. The maximum value is 49.
- **Experience**  
Owned by player character only. The player character will get some number of experience when killing an enemy. The higher the experience, the higher the level of the player.






#### 3.2 Enemy Types

In this game, there are 3 types of enemies, namely:

- **Assaulter**  
Has an aggressive nature. They will go forward as a front-liner enemy. Brings a sword weapon to attack from melee range. This enemy has the characteristics of strong attacks and having a medium amount of life.
- **Archers**  
Have an aggressive nature. Shoots arrows from a distant, hidden place. This enemy has the characteristics of strong attacks, high-speed movement, but a little life.
- **Boss**  
Has a very aggressive nature. Boss has a strong attack characteristics, having a large amount of life, but the slow movement speed.

Here is a table of the enemy types in the game, in Table 1.

**Table 1.** Enemy Types

Enemy Type	Unit Name	Figure
Assaulter	Knight	
Assaulter	Paladin	
Archer	Crossbowman	
Archer	Artemis	
Boss (Mission 1)	High-Priest	

At the beginning of the every mission, the program will generate enemies with random type in a predetermined position in the map. Every enemy will be given control to be able to move. After an enemy died, in a delay of 30 seconds, it will be resurrected with in the same position, with random type.

### 3.3 Character Equipment

Each character in this game will have equipments, which consists of three categories, namely gloves, armor, shoes. Whenever using a certain equipment, the stats the character will increase / decrease according to the equipment used. Here is a table of existing equipment in this game, in Table 2:

Table 2. Equipments

Name	atk	def	spd	agi
Iron Glove	+10	+10	0	0
Steel Glove	+15	+10	0	0
Dark Glove	+30	-10	0	0
Demon Glove	+40	-15	0	0
White Glove	-10	+30	0	0
Holy Glove	-15	+40	0	0
Iron Armor	0	+25	-5	0
Steel Armor	0	+35	-10	0
Dark Armor	0	-10	+30	0
Demon Armor	0	-15	+40	0
White Armor	0	+30	-10	0
Holy Armor	0	+40	-15	0
Iron Shoes	0	0	+5	+10
Steel Shoes	0	0	+5	+15
Dark Shoes	0	0	+30	-10

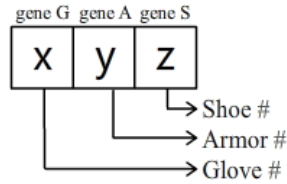
## 4 Genetic Algorithm

In this chapter, will be explained the proposed genetic algorithm process to optimize the set of equipments that will be used in each characters.

### 4.1 Description

Here is a description of a genetic algorithm designed in this game:

- The player character and enemy has complete equipment set (including glove, armor, shoes). The genetic algorithms will perform the auto-equip with the following provisions:
  - For the player character: This equipment can be set manually by player, or player can use the "automatic equip" function, where the program will find the most optimal combination of equipment (with GA) for the player.
  - For the enemy: This equipment is set on each enemy when it first appeared to the map, optimally through the process of genetic algorithm.
- Equipment owned by the player character / enemy marked by a 3 digit of genes, ie genes that represent G (glove), A (armor), S (shoes). These gene numbers are numeric integer between 0-5 because there are 6 kinds of equipment for each type. The gene structure can be seen in figure 1 :



**Fig. 1.** Gene Structure

For example, the gene is written 213, then the equipment used by the character :

- Glove # 2, namely: Dark gloves
- Armor # 1, namely: Steel Armor
- Shoes # 3, namely: Demon shoes

#### 4.2 Fitness Function

The fitness function is used to determine whether a gene is good or not, calculated using the standard deviation function, as explained in chapter 2.3, the implementation is shown in equation (2).

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (2)$$

where:

$$N = 4$$

$$X_1 = (\mathbf{atk}) + (\text{additional } \mathbf{atk} \text{ from equip item})$$

$$X_2 = (\mathbf{def}) + (\text{additional } \mathbf{def} \text{ from equip item})$$

$$X_3 = (\mathbf{spd}) + (\text{additional } \mathbf{spd} \text{ from equip item})$$

$$X_4 = (\mathbf{agi}) + (\text{additional } \mathbf{agi} \text{ from equip item})$$

$$\bar{x} = \text{Mean of } x_0, x_1, x_2, x_3$$

The fitness value is the inverse of the standard deviation values. If the deviation is greater, then the fitness is getting smaller, and vice versa. Fitness value can be formulated based on the standard deviation, shown in equation (3).

$$f(x) = \frac{100}{\sigma + 0.1} \quad (3)$$

With a standard deviation formula used in the calculation of fitness, best fitness is the minimum standard deviation. That is, it is expected that each of the character have a set of equipment that can create equity in character's stats, that is not too high at certain stat, but too low at another stat.

#### 4.3 Genetic Algorithm Process

Suppose an assaulter has the stat: **atk** 12, **def** 15, **spd** 10, **agi** 7. Then, using a genetic algorithm, the program will find the best set of equipment that should be used by the

character. This process is also known as auto equip. The cycle of the GA can be explained below.

1. Generate Random Individuals

There are 10 individuals that will be generated, with random genes.

2. Calculate Fitness

For example, for an individual with gene 445, it means that it equips:

- Glove # 4, namely: White gloves -> -10 **atk**, +30 **def**
- Armor # 4, namely: White armor -> +30 **def**, -10 **spd**
- Shoes # 5, namely: Holy shoes -> -15 **spd**, +40 **agi**

With the base stat of the character is **atk** 12, **def** 15, **spd** 10, **agi** 7, the fitness value can be calculated:

$$\bar{x} = \frac{(12-10) + (15+30+30) + (10-10-15) + (7+40)}{4} = 27.7$$

$$\sigma = \sqrt{\frac{1}{4} ((12-10-27.7)^2 + (15+30+30-27.7)^2 + (10-10-15-27.7)^2 + (7+40-27.7)^2)} = 35.7$$

$$f(x) = \frac{100}{35.7 + 0.1} = 2.79$$

3. Calculate Probability

The greater fitness value, the greater the probability of being selected in the process of roulette. Here is the formula for the probability, shown in equation (4).

$$prob_i = \frac{f(x)_i}{\sum_{i=1}^n f(x)_i} \quad (4)$$

For example, here is the chosen couples that will be copulated:

120 with 120, 120 with 212, 212 with 151, 151 with 212, 101 with 101

4. N-Point Crossover

Crossover is done with chance of 70%. The higher the chance, the higher likeness of couple after copulation.

5. 1-Point Mutation

After the crossover takes place, the result of crossover gene will be incorporated into the 1-point mutation, with chance of 15%. The higher the chance, the higher randomness of genes after mutation.

6. The Process Of Elitism

The process of elitism is the process of replacing the worst gene of the current generation with the best gene from the previous generation, so the gene with best fitness can be maintained.

7. Next Iteration

The process of genetic algorithm aims to find the most optimal set of equipment for a character. The GA will look for the higher fitness as possible. In the process of this GA iteration, the algorithm will stop when:

- No improvement of best fitness value for 3 successive generations, or:



- The number of generations has been more than 10. Limitation is intended that the game is not too heavy during the GA execution.

In Table 3 will be shown individual fitness changes from generation to generation until the genetic algorithm stops.

**Table 3.** Complete Process Example of Genetic Algorithm

	Generation #					
	1	2	3	4	5	6
	445	101	220	125	121	121
	423	120	125	121	125	121
	<b>120</b>	220	251	125	125	321
	212	012	125	125	120	121
	411	100	125	122	121	101
	151	125	100	125	155	121
	212	252	100	125	121	121
	101	111	122	105	121	121
	502	112	101	125	121	101
	553	251	122	225	123	421
Best fitness	8.35	8.54	8.75	8.93	8.93	8.93
Best individual	120	125	122	121	121	121

In Table 3 shown that the genetic algorithm stops at the 7th generation, because of the fitness results have not improved during 3 successive generations. Therefore, can be concluded that the 024 is the best genes.

Conclusion:

For the character who has base stats of **atk 12, def 15, spd 10, agi 7**, the best equipment for it, is a representation of genes 121, namely:

- Gloves # 1, namely: Steel gloves
- Armor # 2, namely: Dark Armor
- Shoes # 1, namely: Steel shoes

## 5 Genetic Algorithm Results

After the genetic algorithm execution for several cases, the results will be documented in this chapter, divided by 2 categories, for the player character, and the enemy character.

### 5.1 Player Character

For the player character, the genetic algorithm is used for auto-equip menu, ie to automatically set the best equipment for player. In this section, the auto equip will be tested for 6 cases, as shown in Table 4. The best equipment is the set of equipment that produces good equalization status.

**Table 4.** GA result for player character

Base stats (with default equipment*)	Auto equip result	Stats after <b>auto equip</b>
<b>attack</b> = 60 <b>defense</b> = 60 <b>speed</b> = 25 <b>agility</b> = 25 (SD = 17.5)	- Iron gloves - Dark armor - Steel shoes	<b>attack</b> = 60 <b>defense</b> = 25 <b>speed</b> = 60 <b>agility</b> = 30 (SD = 16.3)
<b>attack</b> = 60 <b>defense</b> = 60 <b>speed</b> = 100 <b>agility</b> = 20 (SD = 28.3)	- Steel gloves - Steel armor - Steel shoes	<b>attack</b> = 65 <b>defense</b> = 70 <b>speed</b> = 95 <b>agility</b> = 25 (SD = 25.1)
<b>attack</b> = 35 <b>defense</b> = 60 <b>speed</b> = 25 <b>agility</b> = 90 (SD = 25.1)	- Dark gloves - Steel armor - Demon shoes	<b>attack</b> = 55 <b>defense</b> = 50 <b>speed</b> = 55 <b>agility</b> = 65 (SD = 5.4)
<b>attack</b> = 100 <b>defense</b> = 60 <b>speed</b> = 25 <b>agility</b> = 25 (SD = 30.9)	- Iron gloves - Dark armor - Steel shoes	<b>attack</b> = 100 <b>defense</b> = 25 <b>speed</b> = 60 <b>agility</b> = 30 (SD = 29.9)
<b>attack</b> = 35 <b>defense</b> = 100 <b>speed</b> = 25 <b>agility</b> = 25 (SD = 31.3)	- Dark gloves - Demon armor - Holy shoes	<b>attack</b> = 55 <b>defense</b> = 40 <b>speed</b> = 50 <b>agility</b> = 55 (SD = 6.1)
<b>attack</b> = 40 <b>defense</b> = 60 <b>speed</b> = 80 <b>agility</b> = 80 (SD = 16.6)	- Dark gloves - Holy armor - Iron shoes	<b>attack</b> = 60 <b>defense</b> = 55 <b>speed</b> = 70 <b>agility</b> = 80 (SD = 9.6)

\* Default equipment is Glove #1, Armor #1, and Shoes #1

## 5.2 Enemy Character

The auto equip for enemy character is done at the first time it appeared to the map. The results of genetic algorithm for enemy is shown at table 5.

**Table 5.** GA result for enemy character

Base stats (with default equipment*)	Auto equip result	Stats after <b>auto equip</b>
<b>attack</b> = 14 <b>defense</b> = 53 <b>speed</b> = 11 <b>agility</b> = 16 (SD = 17.2)	- <i>Steel Glove</i> - <i>Dark Armor</i> - <i>White Shoes</i>	<b>attack</b> = 26 <b>defense</b> = 18 <b>speed</b> = 31 <b>agility</b> = 36 (SD = 6.6)
<b>attack</b> = 15 <b>defense</b> = 59 <b>speed</b> = 12 <b>agility</b> = 16 (SD = 19.4)	- <i>Steel Glove</i> - <i>Demon Armor</i> - <i>White Shoes</i>	<b>attack</b> = 30 <b>defense</b> = 24 <b>speed</b> = 27 <b>agility</b> = 46 (SD = 8.5)
<b>attack</b> = 17 <b>defense</b> = 44 <b>speed</b> = 14 <b>agility</b> = 21 (SD = 11.8)	- <i>White Glove</i> - <i>Dark Armor</i> - <i>White Shoes</i>	<b>attack</b> = 12 <b>defense</b> = 29 <b>speed</b> = 34 <b>agility</b> = 41 (SD = 10.7)
<b>attack</b> = 19 <b>defense</b> = 44 <b>speed</b> = 14 <b>agility</b> = 28 (SD = 11.4)	- <i>Iron Glove</i> - <i>Iron Armor</i> - <i>Steel Shoes</i>	<b>attack</b> = 38 <b>defense</b> = 44 <b>speed</b> = 14 <b>agility</b> = 33 (SD = 11.2)
<b>attack</b> = 18 <b>defense</b> = 55 <b>speed</b> = 16 <b>agility</b> = 20 (SD = 16)	- <i>Iron Glove</i> - <i>Dark Armor</i> - <i>White Shoes</i>	<b>attack</b> = 35 <b>defense</b> = 20 <b>speed</b> = 36 <b>agility</b> = 40 (SD = 7.6)

\* Default equipment is Glove #1, Armor #1, and Shoes #1

## 6 Conclusion And Suggestion

### 6.1 Conclusion

From the results shown in chapter 5, can be seen that genetic algorithm can be applied to optimize the equipment used by the player and the enemy, based on the stats of each player and the enemy. Each player and the enemy has the status of attack, defense, speed, agility. Genetic algorithm will find a combination of equipment (glove, armor, shoes) the best that can make stats that doesn't deviate too much. The fitness value used in the genetic algorithm is the invers of standard deviation formula. The higher the standard deviation, the lower the fitness, and vice versa.

### 6.2 Suggestion

Here are some suggestions to develop this research to be better:

- Add more enemy types and equipment types in the game so that searching space for genetic algorithm is more extensive, so the genetic algorithm can be more useful.
- Also use the sum of stats for the fitness function, not just standard deviation, because players usually also consider the stats number when equipping character.

## References

1. Ashlock, Daniel. 2004. "Evolutionary Computation for Modeling and Optimization". Italia: University of Venice. p.9
2. Basuki, Achmad. 2003. Algoritma Genetika. Retrieved March 25, 2013 from <http://www.pens.ac.id/~basuki/lecture/AlgoritmaGenetika.pdf>.
3. Janssen, Cory. (n.d.). *Role-Playing Game*. Retrieved December 4, 2014 from <http://www.techope-dia.com/definition/27052/role-playing-game-rpg>
4. Johnson, D dan J. Wiles. 2001. "Computer Games with Intelligence". Australian Journal of Intelligent Information Processing Systems. h.61.
5. Lin, Chih-Sheng, Chuan-Kang Ting. 2011. "Emergent Tactical Formation Using Genetic Algorithm in Real-Time Strategy Games". Conference on Technologies and Applications of Artificial Intelligence. h.325
6. Math Centre. (2003). *Variance and standard deviation (ungrouped data)*.
7. Mitchell, Melanie. (1996). *An Introduction to Genetic Algorithm*. London: MIT Press.
8. Sivanandam, S.N dan S.N. Deepa. 2008. "Introduction to Genetic Algorithms". New York : Springer. h.22
9. Vivek, G and R.C. Narayanan. (2013). *Minimization of Test Sequence Length for Structural Coverage Using VLR*. IJRIT International Journal of Research in Information Technology, Volume 1, Issue 5, May 2013, p. 220-227